1-2021

Towards Effective Detection of Botnet Attacks using BoT-IoT Dataset

Subiksha Srinivasa Gopalan
ssg5920@rit.edu

Towards Effective Detection of Botnet Attacks using BoT-IoT Dataset

by

Subiksha Srinivasa Gopalan

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Networking and System Administration

Department of Computing Sciences

Rochester Institute of Technology

RIT Dubai

January 2021

Committee Approval

Dr Ali Raza — Date
Professor of Computing Sciences
Thesis Advisor

Dr Muhieddin Amer — Date
Professor and Chair
Electrical Engineering and Computing Sciences

Abstract

In the world of cybersecurity, intrusion detection systems (IDS) have leveraged the power of artificial intelligence for the efficient detection of attacks. This is done by applying supervised machine learning (ML) techniques on labeled datasets. A growing body of literature has been devoted to the use of BoT-IoT dataset for IDS based ML frameworks. A few number of related works have recognized the need for a balanced dataset and applied techniques to alleviate the issue of imbalance. However, a significant amount of related research works failed to treat the imbalance in the BoT-IoT dataset. A lack of unanimity was observed in the literature towards the definition of taxonomy for balancing techniques. The study presented here seeks to explore the degree to which the imbalance of the dataset has been treated and to determine the taxonomy of techniques used. In this thesis, a comparison analysis is performed by using a small subset of an entire dataset to determine the threshold sample limit at which the model achieves the highest accuracy. In addition to this analysis, a study was conducted to determine the extent to which each feature of the dataset has an impact on the threshold performance. The study is implemented on the BoT-IoT dataset using three supervised ML classifiers: K-nearest Neighbor, Random Forest, and Logistic Regression. The four principal findings of this thesis are: existing taxonomies are not understood and imbalance of the dataset is not treated; high performance across all metrics is achieved on a highly imbalanced dataset; model is able to achieve the threshold performance using a small subset of samples; certain features had varying impact on the threshold value using different techniques.

Table of Contents

List of Tables

List of Figures

Structure of the Thesis

This thesis is organized as follows:

Chapter 1 consists of related background concepts and provides the motivation and contribution behind this study.

Chapter 2 reviews published work related to the BoT-IoT dataset.

Chapter 3 provides the methodology used for the analysis of related works.

Chapter 4 details the setup, preprocessing steps, and classifiers used for the experimental evaluation on the BoT-IoT dataset.

Chapter 5 provides a comparative analysis of the experimental results.

Chapter 6 concludes this thesis and provides directions for future research.

Chapter 1

Introduction

Large amounts of sensitive user data are prone to various kinds of internal and external attacks. With the advancement of technology, cyberattacks have become evolved with the sophistication of algorithms [1]. The main targets of cyberattacks are systems that process, store essential data or services that depend on their systems [2]. A novel Intrusion Detection System (IDS) is required for the detection of malicious cyberattacks that pose a security issue. IDS is an intrusion detection tool used to detect and classify attacks, security policy violations, and intrusions automatically at both network and host level infrastructures [1].

The evolving nature of attacks has resulted in the need for significant tuning and alteration by incorporating Machine Learning (ML) to improve the performance of IDS [3]. ML is a branch of Artificial Intelligence (AI) that facilitates computer learning without the need for external programming [4]. ML systems help make predictions by learning from existing data. The ultimate goal of ML is to develop an efficient algorithm that processes input data, generates a prediction with the help of statistical analysis [5]. ML algorithms are classified into two main types: 1) Supervised Learning and 2) Unsupervised Learning.

Supervised Learning requires a well-labeled training dataset containing both normal and attack samples. This is a type of learning where the input and the desired output is provided to the learning model to make future predictions [6]. In a binary classification problem, the true or false labels need to be sufficient in quantity when each data sample has a large number of features used as an input to train the model. The performance of a ML model can be strongly affected by the dataset which is used for training. An imbalanced dataset can lead to a biased classification or prediction [1].

Modern IDS need to leverage the power of AI through ML to achieve optimal performance for accurate prediction and classification of attack types [3]. The training required for these ML

models is directly dependent on the datasets used to train them [7]. Neglected or hidden biases in data or an algorithm can lead to biased predictions thereby affecting the performance of an AI application [8].

In recent years, the proliferation of the Internet of Things (IoT) has been increasing all over the world [9]**.** The number of connected IoT devices has the potential to reach 125 billion by the year 2030 [9]. The integration of such IoT devices with various other technologies, services, and protocols has made the management of IoT networks more complex. This leaves the internet vulnerable to serious cyberattacks and threats endangering the consumer of such devices [9]. Some of the most common attacks in IoT systems include Distributed Dos (DDoS), DoS, ransomware, and botnet attacks [10].

There are several IoT based datasets used for the research of anomaly detection and one such dataset that is talked about the most is the BoT-IoT dataset [11]. The BoT-IoT dataset was developed on a realistic testbed that incorporates simulated and legitimate IoT network traffic along with several types of attacks [11]. This dataset has labeled features indicating the attack flow, category, and subcategory for multiclass classification [12].

In this paper, we focus on the BoT-IoT dataset which was published in 2019. This dataset is both rich in terms of the features and attack types. The goal of this study is to analyze the contributions towards the dataset imbalance.

This thesis is organized as follows. Chapter 1 consists of background concepts related to this study, motivation, and contribution. Chapter 2 reviews published work related to the BoT-IoT dataset. Chapter 3 provides the methodology used for this analysis. Chapter 4 details the setup, preprocessing and classifiers used for the experimental evaluation. Chapter 5 compares the experimental results and Chapter 6 concludes our thesis and provides directions for future research.

1.1 Background Concepts related to this study

1.1.1 Botnets and Cyberattacks in BoT-IoT Background

Botnet is defined as a collection of compromised internet-connected devices exposed to hackers. Botnets are used by cybercriminals to initiate botnet attacks. Botnet attacks usually include DDoS, data theft, credential leaks, and authorized access [13].

Hackers use special kinds of Trojan viruses to gain access and breach the security systems of computers [13]. The cyberattacks implemented in botnet scenarios in the BoT-IoT dataset are as follows:

1) Denial of Service

These are cyberattacks that prevent user accessibility by shutting down the target machine or network. This is done by overloading the target machine with traffic thereby initiating a crash When multiple systems orchestrate an organized DoS attack on on a single target machine then it is called as Distributed Denial of Service Attack (DDoS) The BoT-IoT dataset contains both DDoS and DoS attacks These attacks are initiated by bots. [14].

2) Information Theft

It is a type of group attack where sensitive data is seized by compromising the security of the target machine. There are two subcategories of Information theft attacks: Data Theft and Keylogging included in the BoT-IoT dataset [11]**.**

3) Probing Attacks

Probing attacks are attacks where remote systems are scanned for information about the victim. Probing attacks in the BoT-IoT dataset contains two subcategories namely Service Scan and OS fingerprinting. In service scan, request packets are sent by scanning the system's port services. OS fingerprinting gathers information by scanning the operating system of the remote system and the responses are compared to pre-existing responses [11].

1.1.2 Intrusion Detection Systems Background

Intrusion Detection System (IDS)

Intrusion Detection Systems is defined as a monitoring system that can detect suspicious activity and malicious threats [15]. The objective of IDS is to generate an alert, informing the IT personnel of the presence of a network intrusion. Based on the alert information, relevant actions are taken to rectify the threat [16].

Types of IDS

Based on the placement of sensors, IDS can be divided into two main categories: Host-based and Network-based IDS. [16].

1) **Host-based IDS (HIDS):** This type of IDS is deployed at the endpoints in order to protect the host from internal and external attacks. The visibility of HIDS is restricted to the host machine. HIDS has the ability to inspect processes, scan network traffic to-from the host machine, and examine system logs [15,16].
2) **Network-based IDS (NIDS):** NIDS is deployed in the network to monitor traffic flowing through the protected network. This type of IDS provides a wider perspective to detect attacks but limits the internal visibility of endpoints [15,16].

IDS Detection Methods

IDS detection methods are classified into three categories: signature, anomaly, and hybrid [15].

1) **Signature Detection:** It is a method where known threats are detected with the help of a signature. A signature is generated once the malware is identified and it is then added to the list which is used to test incoming traffic. This technique can be a disadvantage when it comes to detecting unknown attacks [15,16].
2) **Anomaly Detection:** This method of IDS detection monitors the behavior of the system. Future behavior is observed and attacks are detected when there is a deviation from normal behavior [15,16]

3) **Hybrid Detection:** This method is a combination of both signature and anomaly-based detection. It can detect low error rate attacks [15,16].

1.1.3 Artificial Intelligence and Machine Learning Background

AI is a machine that displays cognitive behavior similar to that of a human being. AI is an umbrella term that consists of multiple computer programs that display human capabilities. A computer program that has the ability to learn, self-improve, process information, and predict an output fall under the branch of AI [17].

AI can be divided into three domains namely; Robotics, Cognitive Systems, and Machine Learning.

1) **Robotics:** It deals with direct interaction with human beings in the physical world. Robotics is found to be useful in improving work in our daily life [18].
2) **Cognitive Systems:** It is based in the human world where humans and machines communicate and work together towards a common goal. An example is a communication interface called chatbot [18].
3) **Machine Learning:** It is based in the information world. Machines utilize data to learn and derive meaning in order to make predictions. Deep Learning is a subset of machine learning which deals with multi-layer neural networks. [18].

ML is a branch of AI where a system has the capability to learn and improve based on experience without external programming. ML consists of algorithms or methods that are used to create learning models from data. The prime focus of ML is to enable automatic learning for computers without human assistance in order to make accurate decisions in the future [18].

Machine Learning Types

The three types of ML are supervised, unsupervised, and reinforcement learning [19].

1) Supervised Learning

The most commonly used type of learning is Supervised Learning. In SL, the model is trained on labeled data. The labeled data helps the model to learn the relationship between data points. The system is powerful enough to make predictions after plentiful training. This type of learning can compare the predicted output with the expected output to detect errors. These errors are then rectified by modifying the ML model accordingly [20].

A majority of ML models use Supervised ML techniques. SL helps to process and classify data with the help of machine language. There are two types of SL techniques namely; Regression and Classification [20,21].

Regression helps fit the data and produce a continuous value output based on the labeled input data [22]. and Classification deals with the separation and grouping of data into classes. When the input data is labeled into two distinct classes, it is known as Binary Classification. Grouping data into more than two classes is called Multiple/ Multiclass Classification [21].

In the thesis, three classification based supervised learning algorithms: K-Nearest Neighbor (KNN), Random Forest (RF), and Logistic Regression (LR) are used to train the ML model.

2) Unsupervised Learning

In contrast to SL, Unsupervised Learning (UL) is when the training involved is unlabeled or not classified [20]. This type of learning is advantageous than SL as it can process large datasets without spending time on data preprocessing. Due to the absence of labels in UL, hidden patterns are created from the unlabeled data. The creation of such patterns does not require external input from humans. This makes deployment and development of UL more versatile and functional than SL [17].

3) Reinforcement Learning

Reinforcement Learning (RL) is a type of learning that is inspired by how humans learn. [17]. This learning is based on a trial-and-error reward system as it learns continuously by interacting with new environments. The agent is rewarded for correct predictions and penalized

for wrong answers. The model trains itself based on the reward points gained and this process is repeated when exposed to a new environment. [22].

1.1.4 Steps in Machine Learning Background

The machine learning process can be explained in seven steps. Figure 1 depicts the seven steps in ML with data being the key factor for each step.

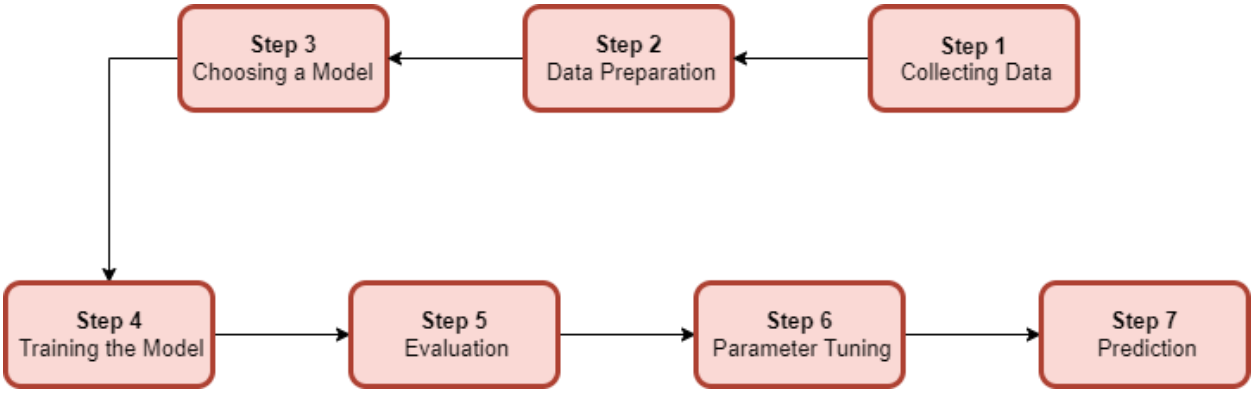

Figure 1. Machine Learning Steps

1) **Collecting/ Gathering Data:** The first step where relevant data is gathered for ML processing. The quality and quantity of data determines the accuracy of the predictive model. [23,24].
2) **Data Preparation:** The raw collected from the previous step is not useful by itself and needs to be cleaned. This is done by removing duplicates, converting data types, and dealing with errors and missing values. Data is visualized to detect outliers, patterns, biases, and relationships between variables [25,26].
3) **Choosing a Model:** There are several models available for different purposes in the field of ML and this step deals with selecting the right model for our goal [25,26].
4) **Training the Model:** Training a model is one the most important steps in ML. It is to improve the model predictions by using the training data. The weights and biases are updated during each iteration cycle also known as the training step [25,26].

5) **Evaluation:** In this step, the trained model is tested on unseen data. The performance of the model is evaluated using performance metrics such as Accuracy, Recall, Precision, etc. A good train-test split is around 70/30 depending on the dataset [25,26].
6) **Parameter Tuning:** This step is otherwise known as "hyperparameter tuning". The parameters are tuned to improve the performance of the model. Simple hyperparameters involve tuning the learning rate, increasing number of training steps, distribution, etc. [25,26].
7) **Prediction:** The final step in ML where the model after undergoing the first six steps is ready to make predictions for practical applications. [23].

1.1.5 Preprocessing in Machine Learning Background

Real-world data generally consists of missing values, noises, and redundancies or available in a format that is not compatible with the ML model. As the capability of a model to learn is directly dependent on the quality of data that is fed as input, data preprocessing becomes a crucial step [27]. Data preprocessing is the process of converting raw data into a machine-understandable format [28].

A dataset consists of a collection of data objects which are also called as samples, vectors, records, events, or cases. These data objects are signified by the number of features. Features describe the characteristics of an object and are also known as attributes, variables, fields, or dimensions. Features of a dataset can be of two types: Categorical or Numerical. Categorical features are those which take on a fixed set of values (eg: Boolean: True or False). Numerical features are values that are continuous or integers [29].

There are various methods of data preprocessing steps available for ML. The five types of data processing methods are as follows:

- **Data Quality Assessment:** Data collected from various sources often contains irregularities in terms of format and quality of information. These irregularities can be due to human error, bugs during the data collection process, or measuring device limitations This becomes a mandatory step that is required when working on any kind of ML problem. There are three techniques to deal with such issues:

1. **Missing Values:** This technique helps estimate missing values and eliminate rows/columns with missing values. In the case of a small percentage of missing values, it is filled using interpolation methods or with the mean, mode, or median values. In the case of a feature consisting of an extremely large number of missing values, that feature itself can be removed. [29].
2. **Inconsistent Values:** Data assessment is performed to correct value inconsistencies (eg: data type of feature) [29].
3. **Duplicate Values:** Redundant values are a common occurrence in datasets and duplicates are removed to avoid bias during the processing of ML algorithms [29].

- **Feature Aggregation:** It is a method of pooling aggregated values to provide a better perspective for data analysis [30]. This helps provide a stable visual of aggregated values than scattered individual values. Feature Aggregation can help reduce the overall processing time and memory consumption [29].

- **Feature Sampling:** Sampling is a technique to select a subset of a dataset for analysis. Large datasets can put a dent in terms of cost, time, and memory constraints, and using a sample size of the dataset can be a better option. Sampling of the dataset is performed by preserving the properties of the original dataset. Feature Sampling can be disadvantageous in the case of an imbalanced dataset where the number of instances of one class is extremely higher than another class [29].

- **Dimensionality Reduction:** Datasets consisting of a large number of features can cause issues during the data analysis phase. It becomes extremely difficult for the model to visualize with the increase in the number of dimensions. Dimensionality Reduction helps convert a high dimensional dataset to a low dimensional one. This is performed using two types of techniques namely Principal Component Analysis (PCA) and Singular Value Decomposition [29].

- **Feature Encoding:** It is the process of transforming/encoding data to an easy and acceptable input for the ML algorithm without compromising the original meaning of the dataset [29].

1.2 Motivation

A considerable effort has been devoted to the research of IDS systems that rely on ML techniques for the development of a dataset. The primary requirement of such systems is to classify events as malicious or benign based on the labeled dataset [31]. The principal issue found during the dataset development stage is the imbalance of datasets. The source of the bias stems from the data that is used for training the ML model [32]. Despite the vast research in ML, only a small percentile of papers discuss the details of the data used for their research. Authors are found to focus more on the construction of complex and accurate models rather than bias in the datasets. The fact remains that the majority if not all big datasets based on ML models are biased [32].

There is a growing body of literature that recognizes the importance of balanced datasets to reduce the bias in ML. Inequality of classes is formed when malicious samples are lower than that of benign samples. As presented in [33,34], imbalanced data causes low detection rates of minority classes where the dataset consisted of more instances belonging to the ethical behavior class than the attack class. A drawback of this is presented in [34] which discusses the performance issues and where ML techniques are proposed as a mitigation approach. It significantly decreases the performance, which results in data loss or overfitting. It intends the trainees to be biased towards the majority class, and all such classes will be identified correctly. Hence, to solve the issues of low accuracy and reliability of classifiers, the balancing of datasets is of prior importance for the identification of sparse classes [35].

The key motivation behind this work is to establish methods in which the impact of bias in a dataset can be minimized when addressing the accuracy.

1.3 Contributions

The key contributions of this thesis can be summarized as follows:

- Provide an analysis of literature for dataset imbalance and identify gaps in existing approaches.
- Provide empirical evidence for the claim that BoT-IoT dataset imbalance is not addressed in cybersecurity research.

- Utilize a real IoT traffic-based dataset for implementation of three supervised machine learning algorithms namely: K-Nearest Neighbor, Random Forest, and Logistic Regression.
- Compare and contrast the results of implementation.

Chapter 2

Related Works

2.1 Related IDS Frameworks in BoT-IoT

There is a plethora of literature on the studies related to ML techniques for intrusion detection systems. This section provides related works on the IDS frameworks proposed and implemented on the BoT-IoT dataset. In addition to the proposed IDS framework, the description of the feature set utilized in the framework is also provided below.

A Hybrid IDS novel ensemble framework is proposed in [36] which combines two types of classifier namely; One Class Support Vector Machine and C5 classifier. The framework is then tested on the BoT-IoT dataset [12] due to the real IoT ecosystem environment representation. The dataset consists of DoS, DDoS, Keylogging, Data Infiltration, OS, and Service Scan attacks. Experimental results show that the proposed framework achieves a high detection and low false-positive rate compared to other techniques.

A Botnet detection method is proposed in [37] which uses a voting system with a hybrid particle swarm optimization (PSO) algorithm. Effective features are detected with the help of the PSO algorithm. Multiple algorithms such as decision tree C4.5, Support Vector Machine (SVM), and deep neural networks were utilized for the identification of botnets. The voting system is also used to classify samples to aid botnet detection. This method was tested on two datasets one of which was BoT-IoT.

A Particle Deep Framework (PDF) is proposed in [10] which is based on network forensics to help with identification of attack in IoT networks. PDF is composed of three functions based on the PSO algorithm to discover anomalous behaviors for a smart home IoT network. This framework is then evaluated on the BoT-IoT dataset and the performance is compared with other

DL techniques. Experimental results show that the proposed framework achieved high performance in terms of accuracy and processing time.

Classification of intrusion attacks is implemented in [38] by comparing the performance of Forward Neural Network (FNN) with a variant of FNN, Self-Normalizing Neural Network (SNN). This analysis is executed on the BoT-IoT dataset as it contains a sufficient amount of records including heterogeneous network profiles. The 10 best feature set, a scaled down version of the full dataset containing 3.6 million records was utilized in this study. The analysis proves that SNN's self-normalizing features make it superior and flexible against opposing samples.

[39] proposed a framework for DoS detection. This framework is composed of training, testing, feature ranking, and data generation modules. It is tested on the BoT-IoT dataset and the results show that the proposed framework achieved a high accuracy compared to other traditional techniques. The BoT-IoT dataset used in this paper contains around 3 million attack 477 normal records.

[40] analysis the performance of seven DL techniques on two models using two real traffic datasets. The two models used for this purpose are unsupervised/generative and deep discriminative models. The 5% train-test version of the dataset is used in this experimentation. The performance for each model is evaluated based on two kinds of classification; binary and multiclass. The performance comparison is done based on three main performance metrics namely; accuracy, false alarm rate, and detection rate

A DeepCoin framework using DL and blockchain-based schemes is proposed in [41]. Hash functions and short signatures were used for the block generation. A recurrent neural network for the detection of fraudulent transactions and attacks in a blockchain network. The DeepCoin framework is tested on three datasets including the BoT-IoT dataset. The CSV files of the dataset are concatenated into one file and then used to form the train and test subset. The three attack types used for this dataset are namely: DoS, information theft, and information gathering. The performance evaluation exhibits the efficiency of the proposed framework.

Security Solutions need to be optimized for scalability to ensure the secure development of IoT. [42] proposes an entropy-based solution to help detect and alleviate DDoS and DoS attacks

in an IoT environment with the help of an SDN data plane. Experimental results exhibit the entropy value correlation of various features for the detection of attacks. It was determined that entropy correlation variation for 4 features: src IP, dest IP, src Port, and dest Port helped in the detection of different attacks. It also exhibits how adding entries to the switch flow table can help SDN mitigation.

[43] proposes a software-based architecture to mitigate and the spread of malware attacks with the help of Network Function Virtualization (NFV). To make it scalable, they have also proposed an RNN-LSTM learning model for the timely detection of attacks. The escalation of malware attacks is monitored with the help of epidemic models. This is followed by patching of systems to contain the damage caused by malware spread. The performance and feasibility of the model are tested on the BoT-IoT dataset due to its detailed characteristics and accurate labeling mechanism. This labeling mechanism is useful in extracting information about the source of the data packet. This information is then utilized for NFV distance-based patching model for IDS.

An IDS based framework is proposed in [1] where DL techniques are applied to classify traffic flow. Binary and Multi-class classification is performed using a feed-forward neural network model. The model is tested on the BoT-IoT dataset which includes attacks such as reconnaissance, denial of service (DoS), distributed denial of service (DDoS), and information theft. The CSV files of the dataset were converted to Apache Parquet files to reduce the overall size of the dataset down to 300MB thereby improving the processing speed. The performance evaluation of this proposed framework exhibits a high accuracy detection rate.

[2] proposes RDTIDS, a novel IDS for IoT networks. Three types of classifiers: decision tree, rule-based concepts, and Forest PA are combined to form RDTIDS. The first two classifiers are used to classify the network traffic of input features as Benign or Attack. The output of the first two classifiers along with the features of initial data as inputs for the third classifier. The model is then tested on two real-traffic based datasets: CICIDS 2017 and BoT-IoT. The 5% train-test version of the entire dataset is used in this study. Experimental results show that the model surpasses other recent ML models in terms of detection rate, accuracy, and false alarm rate.

2.2 Related Data Preprocessing Steps in BoT-IoT

The BoT-IoT dataset consists of 72 million records and preprocessing becomes a critical step required to constrict the size of such a large dataset. Several papers have implemented various preprocessing methods on the BoT-IoT dataset and some of the related works are provided below:

Two preprocessing steps: data transformation and feature selection are performed on the BoT-IoT dataset in [44]. In the data transformation step, class features of the first copy of the dataset are labeled and encoded to binary values, normal and attack traffic as 0 and 1 respectively. The second copy is encoded to 0-9 for representing instances of normal, attack, and nine types of attack traffic. Entropy and Correlation Coefficient techniques are used for selecting 10 features as a feature selection step.

Feature Preprocessing is applied to the extracted data in [45] by removing, encoding, and combining column values. One Hot encoding is applied for categorical data where low dimensional column features and embedding are converted to high dimensional. The preprocessed data is then split into the testing and training sets of data. They are then labeled as normal or malicious traffic and the malicious traffic is further labeled according to the category of attack.

Null/NaN values are replaced with mean or median values by using the impute module from the sci-kit learn package. The most contributed features are determined during the feature selection process. Around ten features were selected from the dataset and the columns are divided into class/labels and features. Categorical features and classes are encoded using the Label Encoding technique to make it more convenient for analysis [46].

A two-step preprocessing is executed on the datasets in [2]. Symbolic valued attributes are mapped to numeric valued attributes as the first step. In the second step of preprocessing, attributes with high numeric ranges are scaled down to lower numeric ranges. This is implemented to avert numerical complications during the calculation process.

Before the data is inputted into the ML model, it is cleaned and formatted. Inconsistent and missing data is cleaned before identifying essential features that can achieve the best output. RNN-LSTM model is implemented in [43] which uses cross-correlation technique for preprocessing. All

characteristics of the data are used for experimentation and the impact is studied with the removal of highly correlated features.

Data is preprocessed in [47] using three steps: data transformation, data normalization, and Up-Sampling. Data Transformation involves dropping rows containing missing values. The label Encoder function from sklearn is used to convert nonnumeric features to numeric values. The output label ‘category’ undergoes hot encoding to prevent performance degradation. Features containing IPV4 and IPV6 addresses and hexadecimal format are converted to numeric values and integers respectively. MinMaxScalar function is applied as a normalization step on feature vectors. This can improve the efficiency of the IDS. Upsampling is implemented on the normal samples to solve the issue of imbalance.

Feature extraction is carried out in [1] using the TShark tool to extract packet information from PCAP files. Generic features of the traffic are captured instead of the attack-oriented features. The extracted information then undergoes feature preprocessing where column values are dropped, merged, and encoded. These results are stored in arrays.

[10] applies a preprocessing step to handle unuseful and missing features. This step also helps produce and rescale new features which can benefit the performance of the ML model.

[48] applied a bijective soft set and proposed metric approach called CorrACC for effectively selecting features. Soft Set helps display the relationship of the statistical features and the most effective features are selected for IDS.

Redundant and irrelevant features are efficiently decreased using a feature selection component. Information Gain (IG) is implemented and features with less IG values are removed and values with IG value indicate features that are useful for differentiating the class [36].

2.3 Importance of dataset balancing

Supervised machine learning (ML) techniques require labelled datasets to train a classification or prediction model. In a binary classification problem, the true or false labels need to be sufficient in quantity when each data sample has a large number of features used as an input

to train the model. The performance of a ML model can be strongly affected by the dataset which is used for training. An imbalanced dataset can lead to a biased classification or prediction [52].

A considerable amount of literature has been published on balancing techniques. Some of the researches have published reviews such as in [49]. It addresses the adverse effects of class imbalance and establishes a computational framework consisting of data level and algorithm level solutions. In [50], the review presents the classification of imbalanced data, evaluation measures, and the class imbalance problem in the presence of multiple classes. This is of significance to IDS where a malicious event should be detected and the type of the event should also be classified, both with high accuracy. A survey on existing approaches on handling classification with an imbalanced dataset is presented in [51].

Previously published studies on the taxonomies of balancing techniques are not consistent. The existing accounts fail to resolve the contradiction between data level, algorithm level, and hybrid level techniques. The definitions for these levels found in [49 - 51], which are seminal review papers, indicate that the generalizability of the proposed taxonomies is problematic. This is because a systematic understanding of how imbalance techniques are defined is still lacking. Most of these studies have largely focused on data level and algorithm level techniques with a lack of clarity in defining hybrid level techniques.

Although studies involving ML in IDS have recognized the importance of balanced datasets, a significant number of works have been done which do not address this issue. Surprisingly, the accuracy of the ML algorithms is presented with seldom studies of imbalance and it is unclear to what extent this has been accounted for. To date, no large-scale studies have been performed to investigate the prevalence of performance results using imbalanced datasets in cybersecurity.

2.4 Taxonomy of balancing level techniques

Existing taxonomies show that there are two approaches when it comes to treating class imbalance classification: Data Level and Algorithm Level. In the data level, a balanced class

distribution is obtained by adjusting the class imbalance ratio. At the algorithm level, learning tasks related to a minority class is improved by fine-tuning existing classification algorithms [51].

The following is a review of the types of balancing techniques collected from three highly cited survey papers.

Data Level

Data level approaches treat the imbalanced class distribution as a preprocessing step [51]. The solutions at data level involve various forms of resampling [50]. Class distribution can be rebalanced by applying either over sampling or under sampling techniques.

1) Under Sampling

Undersampling eliminates samples from a majority class to reduce the disparity between the two classes [51]

2) Oversampling

It is performed by duplicating the samples in the minority class samples to reduce the ratio of imbalance between classes.[51]

SMOTE

Synthetic Minority Over Sampling Technique (SMOTE) is defined as an adaptive form of oversampling [51]. SMOTE is considered to be complex compared to under and oversampling methods and has gained popularity as an important technique when it comes to treating class imbalance [51] In SMOTE, new instances of minority classes are created by appending numerous minority classes that are nearest to each other [50]. The new minority class will ultimately reduce the degree of class imbalance compared to the original imbalance ratio.[49]

Feature Selection

Another preprocessing step besides sampling is feature selection. Feature selection is usually applied by adopting either of the two methods 1) filter method or 2) wrapper method. Filter method acknowledges the intrinsic features to measure the integrity of the feature subset and wrapper method deals with wrapping the induction algorithm with the feature selection process [51]

Algorithm Level

Algorithm level methods are algorithms that are committed to learning directly from the imbalanced class distribution in the datasets. Subcategories of algorithm level techniques include one class cost-sensitive and ensemble learning algorithms.

1) One Class Learning

One class learning is otherwise known as recognition-based learning. It is a method where a classifier is modeled on the minority class samples. Neural networks are applied to learn from the minority class samples instead of identifying patterns from the majority and minority class samples. A key point in this approach for classification is the threshold. A strict boundary threshold will separate the minority class and a lenient threshold blanket the majority class in the boundary.[51]

2) Cost Sensitive Learning

Cost sensitive learning is created with the concept that a classifier is assigned with a high cost based on the type of misclassification. An example of this learning would be when a larger cost is assigned by the classifiers to false negatives than false positives. This will lead to the correct or misclassification of the positive class [51]. A cost matrix is considered during the initial stage of model building to generate a low-cost model [50]. A disadvantage of the cost matrix is that the real cost of several applications including balanced datasets is unknown. In such cases, an artificial cost value is generated [51].

3) Ensemble Learning

The ensemble method is another type of learning to treat the issue of class imbalance. In this method, several classifiers are applied to the training data and a final decision is produced from

the aggregated predictions. The objective of combining multiple classifiers is to enhance their ability to generalize [50].

Ensemble methods can be also described as bagging or boosting. Bagging generates more samples from the original data for the training set to reduce prediction discrepancy. Boosting produces an output based on the model classifiers that are experimented on training data. Some of the most commonly used ensemble learning methods include Bagging, Random Forest, and AdaBoost [51].

Hybrid Level

Hybrid is a new breed of learning where the hybridization of two or more individual methods is designed to alleviate the problem of class imbalance [51]. It is used to deal with a specific part of an overall solution by using multiple algorithms [50]. Problems in subset feature selection, sampling, optimization of the cost matrix are solved by hybridization [51]. When using hybrid approaches, it is important to ensure that the different classifiers used are complementary to each other to yield a high combined performance compared to the performance of an individual method. [49]

2.5 Contentions in existing taxonomies of balancing level techniques

Perhaps the most comprehensive account of existing techniques and an indication of a taxonomy is provided in [49], [50] and [51]. The authors in [49] review the approaches which span over the last 8 years. In contrast, [50] and [51] do not specify the year span of their reviews.

Table 1 derives the taxonomy based on approaches in [49 - 51]. Two important classifications emerge from the studies in [49- 51]: techniques classed as data level; techniques classed as algorithm level. Collectively, these studies converge on the definition of data-level methods to include data sampling and feature selection approaches, while algorithm level methods include cost-sensitive and hybrid/ensemble approaches.

In [49] and [51] the authors define data-level methods to include data sampling and feature selection approaches, while algorithm level methods are defined to include cost-sensitive and hybrid/ensemble approaches. Across all three surveys shown in Table 9, several divergent accounts of algorithm level classifications have been proposed, creating numerous discrepancies.

In [50] the major deviation is in the algorithm-level definition. In contrast to [49] and [51], subcategories of algorithm-level are not defined in [50]. The subcategory of *one class*, however, is mentioned in [54] under the discussion of algorithm-level but not distinctly classified as in [49] and [51].

Table 1. Contentions in Existing Taxonomies

Reference	Title of Paper	Publication Date	Citations	Data Level			Algorithm Level						Cost Sensitive Learning	Boosting Approaches
				Data Level	Feature Selection	Data Sampling	Algorithm Level	Cost Sensitive Learning	Ensemble	Hybrid	One Class	Improved		
[49]	A survey on addressing high-class imbalance in big data	2018	55	☑	☑	☑	☑	☑	☑	☑	☐	☐	☐	☐
[50]	Classification of Imbalanced Data: A Review	2009	788	☑	☐	☑	☑	☐	☐	☐	☑	☐	☑	☑
[51]	Classification with class imbalance problem: A review	2015	157	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐

The derived taxonomies in [49] and [51], which are more recent, show that feature selection is a subcategory of data level whereas [50] does not. We cautiously suggest that this may be because the feature selection approach gained popularity in the study of dataset bias after the publication on [50]. Both [49] and [51] discuss techniques in feature selection such as principal component analysis (PCA) and the likes since the publication of [50]. The specifics of the feature reduction techniques and its development over the years is beyond the scope of this paper. In contrast to [8] which addresses the concept of *Improved learning*, [49] and [50] do not discuss this as a subcategory or part of the taxonomies provided.

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone [52]. In contrast to [51] which defines ensemble as boosting (an iterative technique that adjusts the weight of an observation based on the last classification), [49] defines ensemble as both bagging (a way to decrease the variance in the prediction by generating additional data for training from the dataset using combinations with repetitions to produce multi-sets of the

original data) and boosting. The definition of ensemble provided in [49] aligns better with the definition provided in [52].

Another major deviation observed in [49] and [51] from [50] is the inclusion of ensemble under algorithm level in [49] and [51]. In [50] ensemble, cost sensitive and other boosting are included as subcategories of boosting. This is not shown in Table 9 due to the lack of space. The definition of ensemble provided in [50], however, agrees with the definition provided in [52].

2.6 Literature of balancing techniques used in BoT-IoT dataset

A large and growing body of literature has investigated ML methods applied to cybersecurity datasets. At the time of this study, 28 papers were published between 2019 and 2020 on ML which uses the BoT-IoT dataset. This could be due to the recent publication of the dataset. Out of the 12 papers, a total of 3 papers were found to have treated the imbalance using only algorithm level techniques in the BoT-IoT dataset.

Class weights are the most common balancing technique used in the BoT-IoT dataset. Class weights in [53] are used on the training dataset to resolve the issue of imbalance. The weights are calculated based on the inverse of quotient value by dividing the packet count of a particular class by the packet count(maximum) of all classes.

Another paper that incorporates class weights is [10]. They have reservedly mentioned the implementation of class weights to treat the imbalance with no explanation provided on the process or technique.

An ensemble of classifiers is used as an algorithm level technique to reduce the bias in [2] where a novel IDS framework, named RDTIDS is proposed. RDTIDS uses two types of classifiers (binary and multiclass) in parallel that feeds to a third classifier thereby minimizing the dataset imbalance. The classifiers used for this purpose combine different approaches based on rule-based concepts and decision trees.

There have been no papers found to be related to the BoT-IoT dataset that has used either data level or hybrid level techniques. A couple of papers mentioned imbalance but not with regards to our study of the BoT-IoT dataset.

Chapter 3

Literature analysis of balancing techniques in BoT-IoT

3.1 Overview of BoT-IoT dataset

BoT-IoT, an IoT dataset was created by the Cyber Range Lab of The Center of UNSW Canberra Cyber and was designed on a realistic testbed environment [12]. The dataset environment contains both simulated and genuine IoT attack traffic. The traffic is generated by applying six types of attacks in five IoT devices. This recorded network and normal traffic are extracted to form the BoT-IoT database [37]. It is a well-structured and dataset for IoT network forensic analytics [42]. BoT-IoT contains sufficient amounts of records for heterogeneous network profiles which was stimulated with the help of the Node red tool [38]. This dataset uses a lightweight protocol (MQTT protocol) making it applicable for various IoT solutions. Five IoT scenarios were used to trigger legitimate IoT traffic [11].

The source files for this dataset are provided in various formats such as pcap, argus, and CSV files. To further aid the labeling process, the files are classified into attack categories and subcategories. The BoT-IoT dataset contains more than 72 million records composed of 74 files with each row containing 46 features. The dataset includes various attacks such as keylogging, data infiltration, OS, service scan, DoS, and DDoS. The authors have also extracted 5% of the original dataset to ease the computational processing of the datasets. This 5% dataset contains 4 files with approximately 3 million records [12].

3.2 Imbalanced Class Distribution of BoT-IoT dataset

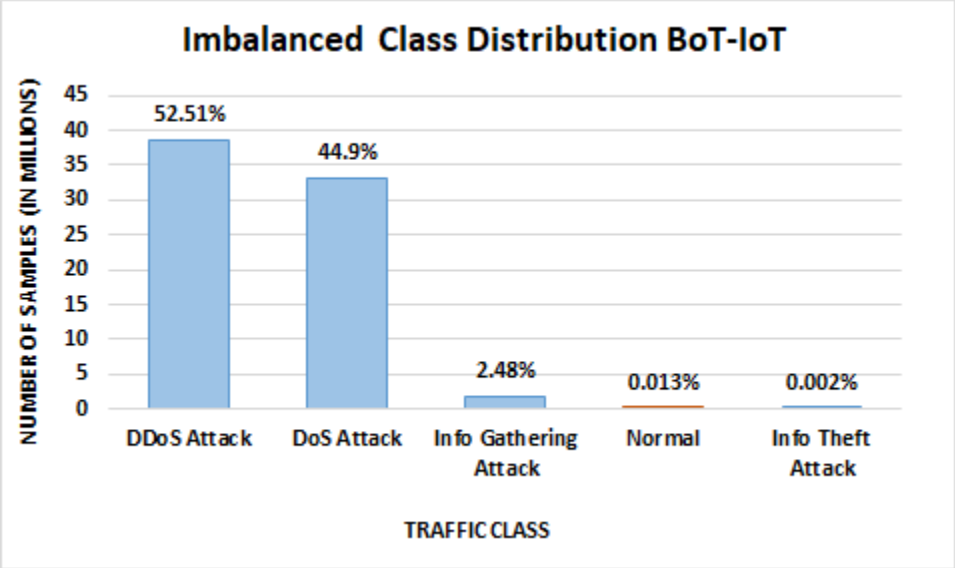

Figure 2. Imbalanced Class Distribution for BoT-IoT

Figure 2 highlights the benign traffic to malicious traffic imbalance observed in the BoT-IoT dataset. An exponential trend is observed when evaluating the distribution of samples per class in the BoT dataset. The trend indicates that the percentage of malicious samples is higher than benign samples. The majority classes in the BoT-IoT dataset are attack types while the normal traffic is part of the minority classes. The ratio of benign to malicious for the BoT-IoT dataset is 1:7687. A dataset best serves in machine learning algorithms when the distribution is normal or close to normal.

In Table 2, the individual flow count distribution derived from [2] for various types of attacks is shown. The number of samples for DDoS and DoS attack is extremely high compared to the benign samples. This dataset would be better served to distinguish between a DDoS and DoS attack as they both have relatively the same amount of samples.

The issue of class imbalance distribution is dominant across all domains [54]. In this paper, two metrics - precision and recall - are studied and referenced for issues that relate to class imbalance. A dataset contains imbalanced class distribution when one class - which is often the

one of interest - is a minority class or not represented adequately. Several publications have been reviewed to determine the impact of dataset imbalance. By far the most widely accepted account can be found in [54] which states that issues arising from this are poor accuracy in classification and a general bias in the results obtained. Classifier algorithms used in machine learning as mentioned in [55] require a balanced dataset. Despite these findings, there are recent developments in algorithms that account for the imbalance in a dataset. Such approaches are generalized as algorithm level approaches towards the mitigation of bias [51].

Table 2.Class Distribution for BoT-IoT

Class Label	Count
DDoS	38,532,480
DoS	33,005,194
Information Gathering	1821639
Normal	9,515
Information Theft	1,587
Total	73,370,443

Table 3. Imbalance Ratio Distribution for BoT-IoT

Normal Traffic Count	Attack Traffic Type	Count	Normal to Attack Ratio
9515 (0.013%)	Information Gathering	1821639 (2.48%)	1 *to* 191
	DDoS	38532480 (52.51%)	1 *to* 4038
	DoS	33005194 (44.9%)	1 *to* 459
	Information Theft	1587 (0.002%)	6 *to* 1

In Table 3, we provide an approximation of ratios of benign to malicious traffic in the various attack types (majority class) as provided in Table 4 from [2]. Due to the high volume of malicious traffic in the BoT-IoT dataset, evidence in literature [2,37,53] has established that achieving acceptable True Negative (i.e., where the benign traffic is classified as benign) accuracy is questionable. This is due to the bias towards classifying benign traffic as malicious. DDoS and DoS attacks have a ratio of 1:4038 and 1:3459 respectively which is extremely imbalanced. Information Gathering has a ratio of 1:19. The above cases will yield a high rate of false positives (i.e., where benign traffic is classified as malicious). In the case of Information Theft, the number of benign samples is more with a ratio of 6:1. This will lead to a high amount of false negatives.

3.3 Criteria for BoT-IoT Paper Selection

Our approach to deriving a new taxonomy is based on the study of work published in the cybersecurity domain anchored on the BoT-IoT dataset. A criteria for selecting published work for this dataset, related to AI-based intrusion detection systems were developed. The criteria used for selecting papers related to IDS and dataset balancing is specified in this section as follows:

Selection Criteria for IDS related papers were as follows:

- Publications were only included if they were relevant to the BoT-IoT dataset

- Publications were only included if certain keywords and phrases related to dataset balancing were found. These included but were not limited to: imbalance; minority class(es); majority class(es), sampling, downsampling, upsampling, balance(ing).
- Publications were only included if they were published between 2019 and 2020 to align with the first public announcement of the dataset.
- The number of citations and venue of publication were considered for each work assessed.
- Publications were only included if they approached imbalance and cited the other non-IDS related work on dataset balancing.

To come up with this methodology, previous approaches published in [56] and [57] were reviewed and analyzed. The two papers were compared, in [56] more advanced techniques were proposed due to the recent advancements provided by the google scholar platform reported in this paper.

From [56], google scholar as a platform provides access to key information about the citation of a paper. The name of the primary dataset paper is first entered in the google scholar search engine. The number of times the paper has been cited is displayed and clicking on it leads to the total list of cited papers. The papers are filtered down by clicking the checkbox "Search within cited articles", the keyword search and custom range process used for the datasets are explained below:

The keyword "imbalance" and "balance" was used for the BoT-IoT dataset and a total of 12 papers were generated.

A custom range option is also available in google scholar to select the key papers. The range applied for our research is *2019-2020.*

3.4 Analysis of Published Works in BoT-IoT

A systematic review of the literature in the cohort of published works from Section III allowed us to divide the work into groups according to the level of contribution each work makes towards balancing of a dataset. This grouping has been done in a hierarchical order as shown in Figure 3 with the first level determining whether the paper has a contribution or not. The second level identifies the extent of the contribution in terms of a proposed method or the application of an

existing method. In the case of non-contributing work, the second level identifies whether imbalance has been recognized and mentioned or not.

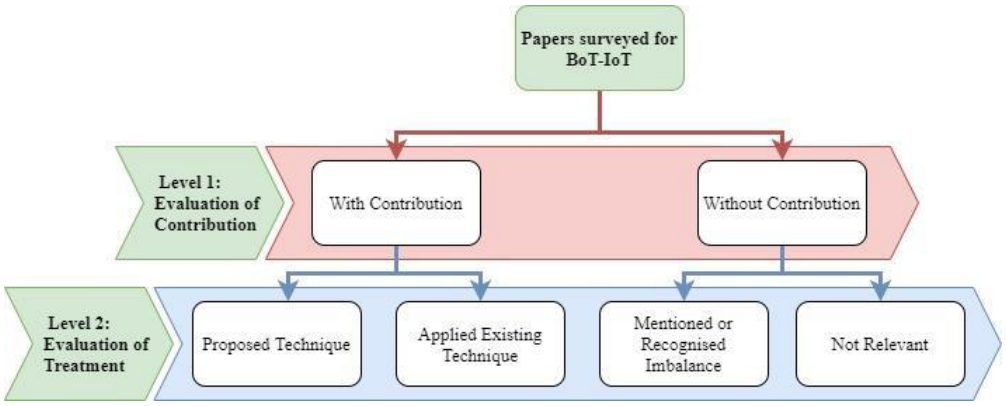

Figure 3.Hierarchical grouping of published work

A more thorough definition of the grouping has been provided below:

- **Proposed:** The authors have proposed a technique to solve the imbalance.
- **Applied Existing:** The authors have applied existing techniques in solving imbalance.
- **With Contribution:** This is a cumulative of papers in which the authors have either proposed or applied existing techniques.
- **Mentioned:** These are the papers in which the authors have mentioned an imbalance technique with respect to our analysis but have not treated it.
- **Not Relevant:** The authors have mentioned imbalance in general and not with respect to our analysis of dataset imbalance.
- **Without Contribution:** The authors of these papers have either mentioned imbalance or did not have any discussion relevant to the imbalance of the datasets.

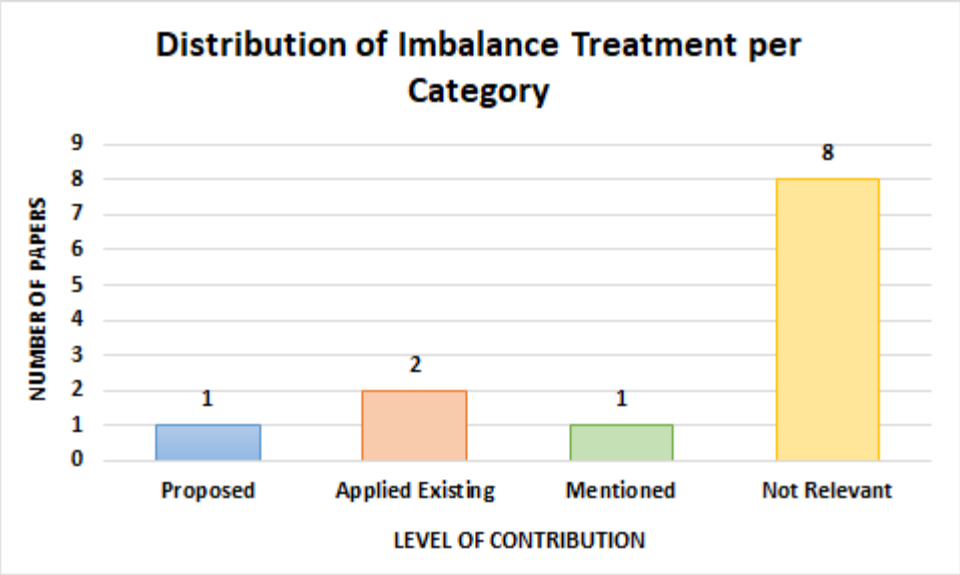

Figure 4. Comparison of the imbalance treatment categories across BoT-IoT dataset

The categories: *with contribution* and *without contribution* have been added for the ease of presenting the analysis. Papers that have proposed a new technique or applied an existing technique to deal with the imbalance are labeled as *With Contribution*. Furthermore, papers that have no relevance to dataset imbalance or have reservedly mentioned the word imbalance are collectively labeled as *Without Contribution*.

Figure 4 shows the distribution of published work across the four groups defined for the BoT-IoT dataset. Figure 5 presents a cumulative percentage distribution across the four groups irrespective of the datasets used. It has been observed that only 25% of the papers have either proposed or applied techniques to treat dataset imbalance and the remaining 75% of the papers have not contributed to this study. These results further support the idea that there is a lack of attention to imbalance because 67% of the papers are *not relevant* which takes precedence over other groups.

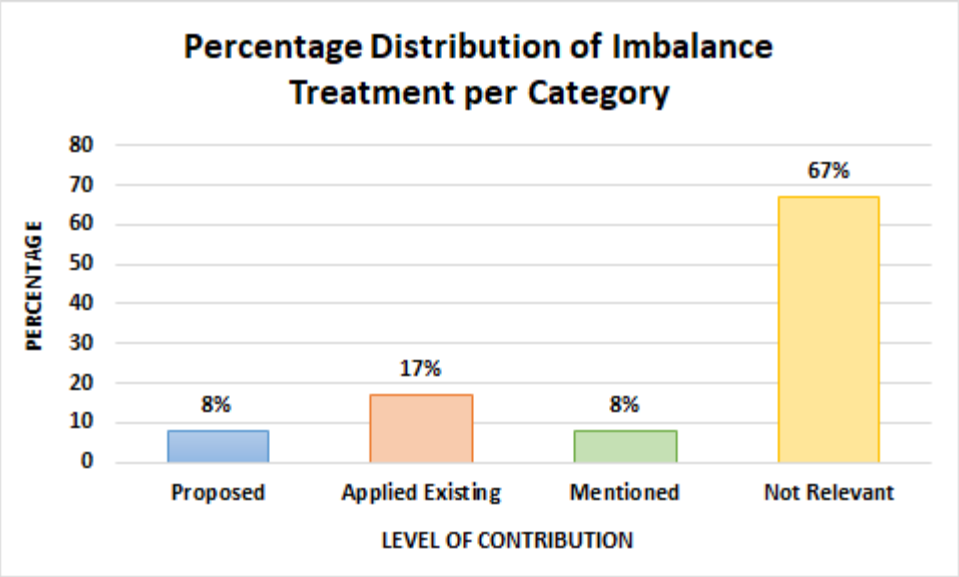

Figure 5. Categorization of Imbalance Treatment- A comparison of all papers surveyed, across all categories of imbalance treatment.

The "*without contribution"* category shown in Figure 4 takes precedence with the highest count. In the "*with contribution"* category the *applied existing* takes precedence as shown in Figure 6.

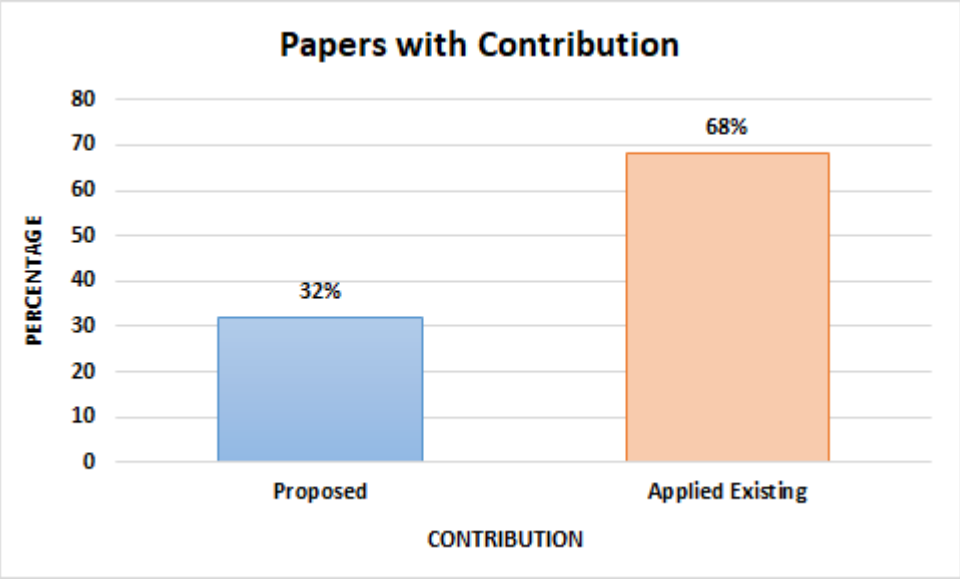

Figure 6. Comparison of papers in which proposed techniques are compared with existing techniques that have been applied.

3.5 Level Distribution of BoT-IoT dataset

A study of the proposed and applied existing papers was undertaken to determine the technique which has been used. In contrast to the findings in the literature review, the outcome of this particular study demonstrates that there are three distinct classifications of techniques for balancing of datasets as shown in Figure 7. The grouping or classification of these techniques are defined as levels to be consistent with published literature presented in Chapter 2.

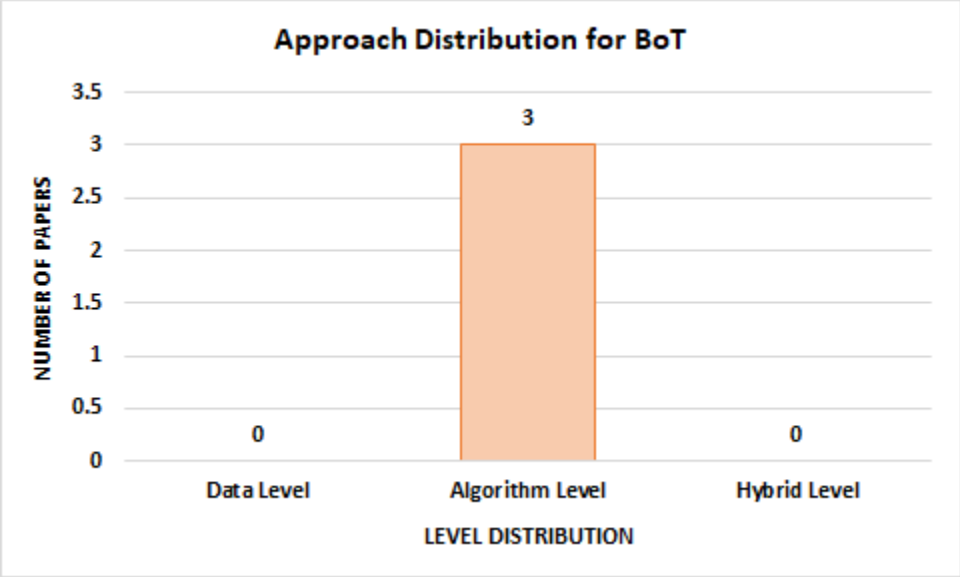

Figure 7. Comparing the number of papers that propose, apply or mention approaches across all three datasets

The statistical representation in Figure 7 spans the 3 papers out of 12 papers published in the BoT-IoT. This dataset shows that all 3 papers have used algorithm level techniques to solve the issue of imbalance. There is no relevant study performed using data level and hybrid level techniques for the treatment of imbalance in the BoT-IoT dataset.

3.6 Ranking of approaches

The percentage distribution for *proposed* and *applied existing* from the total amount of papers *with contribution* is depicted in Figure 6. Papers with contribution had 32% of new methods proposed and 68% of existing methods applied. This result may be explained by the fact that a majority of the papers have not focused on proposing a new technique to overcome bias. This discrepancy can be a dominant focus area for future research directions.

Chapter 4

Setup and Preprocessing Steps

4.1 Setup

All stages of this experiment were performed using Jupyter Notebook on a Lenovo Yoga Laptop with a Windows 10 64-bit operating system. The processor was an Intel Core i5 with processing unit of 1.60 GHz and a RAM of 8.0 GB. Pandas and NumPy library were used to load and perform preprocessing on the dataset. Data visualization was done using Matplotlib library and Scikit-learn was used for training-testing data, model evaluation and performance evaluation metrics.

The details of the server used in this implementation are as follows:

- HP ProLiant DL380p Gen8
- CPU: Dual Intel® Xeon® CPUnE5-2660 v2 @ 2.20GHz (40 cores total)
- Memory: 256 GB ECC (1866 MT/s)
- Storage: 9TB
- Operating System: Linux (Fedora release 32)

4.2 Dataset used for this study

The dataset used for this experimentation is the BoT-IoT dataset developed by the UNSW, Canberra [12]. The full dataset contains 72 million records and including normal traffic in addition to a variety of attack traffic such as DDoS, DoS, Data exfiltration, OS, and Service Scan [40]**.** The statistical distribution and description of the dataset are provided in Chapter 3. The CSV version of the BoT-IoT is provided in the form of 74 separate files. For ease of handling, the traffic for two attacks: DDoS and Reconnaissance along with Benign traffic were considered for the experimental evaluation.

4.3 Preprocessing Steps

The 74 CSV files of the BoT-IoT dataset are concatenated into one single file before applying the preprocessing steps. The following data preprocessing steps were carried out on the BoT-IoT dataset:

1) **Add Feature Names:** The feature names are added to each column of the dataset.
2) **Dropping of Empty columns:** Six completely empty column features: smac, dmac, souit, doui, sco, dco' are deleted from the dataset.
3) **Replace Empty Ports with 0:** Two features 'sport' and 'dport' containing a small percentage of NA values were filled with 0.
4) **Replace port values with 0:** Port values of 'sport' and 'dport' are replaced with 0.
5) **Converting data type:** Data types of 'sport' and 'dport' columns are converted into an integer value.
6) **Dropping non-required features:** Features that are not required for the ML model are removed from the dataset.
7) **Encoding object to Categorical Value**: Three features 'flgs', 'proto' and 'state' are factorized to encode the object as a categorical variable.

The preprocessing steps implemented on the BoT-IoT dataset resulted in reducing the number of features from 35 to 25. The preprocessed data is then used to train the ML model.

4.4 Supervised ML classifiers used in this study

Three types of classification based Supervised ML techniques were used in this study: 1) K-Nearest Neighbor (KNN), 2) Random Forest (RF) and 3) Logistic Regression (LR)

4.4.1 K-Nearest Neighbor

KNN is a simple and easy to implement ML algorithm where data is classified based on similar distance measures. The distance, in this case can either be a Manhattan or Euclidean type. The data points distance nearer to each other is calculated and the value of k is selected [58]. The value of k can be any integer value and data points of nearest distance are assigned the same

class. The accuracy of the model increases with the increase in the number of nearest neighbors (i.e., value of k) [59]. KNN algorithm is nonparametric as it does not form an assumption about the data which can be advantageous in the case of real-world data.

4.4.2 Random Forest

Random Forest is based on an ensemble learning concept where multiple classifiers are combined to solve complex data problems and to improve the overall performance of the model. RF consists of numerous individual decision trees that work on different subsets of the provided dataset. The vote from each tree is taken into account and the final output is predicted based on the majority votes. An increase in the number of trees in RF can lead to an increase in accuracy performance thereby avoiding the issue of overfitting [60].

4.4.3 Logistic Regression

It is a method to estimate distinct values based on an input set of independent variables. The predicted output for the algorithms is a dependent categorical variable. It helps predict a probabilistic value that ranges from 0 to 1. LR is similar to Linear Regression but differs when it comes to the type of problem that is needed to be solved. LR is significant compared to other ML algorithms as it has the capability to classify a diverse type of data and determine the most useful variable [61].

4.5 Machine Learning Validation Metrics

In addition to data preparation and training of ML models, performance evaluation is also a key step. The performance of the ML model is evaluated with the help of different performance metrics based on the confusion matrix. It is important to determine the overall performance of the model before testing it on new unseen data [62].

4.5.1 Confusion Matrix

Confusion Matrix is a performance measurement technique for machine learning classification. The performance is measured with the help of true values of the testing data. Confusion Matrix

helps in detecting errors (FP and FN) and also calculating various performance metrics values in ML [63]. The confusion matrix in Table 4, comprises of four items for binary classifiers:

Table 4.Confusion Matrix

Confusion Matrix		Actual Values	
		Positive	Negative
Predicted Values	Positive	TP	FP
	Negative	FN	TN

True Positives (TP): when the classifier identifies the true positive label as positive

True Negatives (TN): when the classifier identifies the true negative label as negative

False Positives (FP): when the classifier identifies the true negative label as positive

False Negatives (FN): when the classifier identifies the true positive label as negative

In the context of cybersecurity research, a well-known understanding is that a positive event is defined as a malicious event and the correct classification of such an event is deemed as a true positive outcome. A negative event is a benign event and the correct classification is deemed as true negative. Inaccurate classification can mean that a benign event is classified as a malicious event. This misclassification is deemed as a false positive. Likewise, for a malicious event to be classified as a benign event is deemed a false negative [26].

4.5.2 Performance Metrics

There are various metrics used to evaluate the performance of a ML classifier. The performance metrics are evaluated based on the four values (TP, TN, FN, and FP) of the confusion matrix. The metrics used in this thesis are as follows:

1. **Accuracy:** This metric helps calculate the accuracy of a classifier. It determines the number of positive predictions made by the model. It is the ratio of the number of positive predictions over the total number of predictions made by the model.

$$\text{Accuracy} = \frac{\text{TP+TN (Number of positive predictions)}}{\text{TP+FP+FN+TN (Total number of predictions)}} \quad [64]$$

2. **Precision:** It is the ratio of the number of positive predictions by the total number of positive values predicted by the classifier [64].

$$\text{Precision} = \frac{\text{TP (Number of positive predictions)}}{\text{TP+FP (Total number of positive values predicted by the classifier)}} \quad [64]$$

3. **Recall / Sensitivity:** It is calculated by the total number of true positive predictions divided by the sum of all samples that belong to the positive class.

$$\text{Recall} = \frac{\text{TP (Number of positive predictions)}}{\text{TP+FN (Sum of samples belonging to the positive class)}} \quad [64]$$

4. **F1 Score / F-measure:** It helps to calculate the precision and recall simultaneously by generating the harmonic mean of the two metrics. Equal values of recall to precision can achieve maximum F1 score.

$$\text{F1 Score} = \frac{\text{2X(Recall X Precision)}}{\text{Recall + Precision}} \quad [64]$$

Chapter 5

Experiment and Analysis

5.1 Experiment 1: Modeling on the full dataset using train and test split

In the first experiment of this study, the model is trained on the BoT-IoT dataset by using the train and train split function from scikit learn library. Since the dataset contains around 72 million records, for the ease of handling, two attacks: DDoS (38 million samples) and Reconnaissance (around 1 million samples) were considered for this study. The ML classifiers used in this experiment are KNN, RF, and LR.

Table 5.Classifier accuracy using scikit learn train and test split

	Attack Type	Classifier	Classifier Score
Benign 9,515 samples	**Reconnaissance** 998,007 samples	KNN	0.9983319019
		RF	0.9999784868
		LR	0.9963262125
	DDoS 38,532,480 samples	KNN	0.9999860837
		RF	0.9999848485
		LR	0.9997626359

As observed in Table 5, the results obtained show that the model was able to achieve a performance of 99% for both attacks for all three classifiers despite the dataset being highly imbalanced (Chapter 3). The number of false positives and false negatives obtained for both attacks were extremely small. Overall, the model was able to achieve exceptional results by using a large number of samples and this analysis gave birth to the objective for the next experiment.

5.2 Experiment 2: Threshold Analysis for Reconnaissance and DDoS Attack

As shown in Table 5, the model was able to achieve high performance using large number of records with a highly imbalanced class distribution. This experiment aims to discover the threshold sample limit at which the model attains the highest performance by using a small subset of samples from the entire dataset.

The BoT-IoT dataset consists of an exceptionally large number of malicious samples compared to benign samples (Chapter 3) and this can put a strain on cost, memory and time consumption. Similar to the previous experiment, two attacks: DDoS and Reconnaissance were considered for this study. Initially all 74 CSV files of the BoT-IoT is concatenated into a single file. The above-mentioned attack traffic along with normal traffic is extracted from the concatenated file. This step is followed by feature selection preprocessing where a small subset of samples is taken for the implementation: 10,000 and 1,000 samples of malicious and benign traffic respectively. Preprocessing steps are then applied on the subset of samples (Chapter 4).

A series of ten iterations is performed by manually entering the number of training samples for each iteration. The iteration samples were chosen by a trial-and-error method. The first occurrence of the highest value in the series of ten iterations is treated as the **Threshold/Breakpoint** value.

This experiment is broken down into two parts. In the first part, the number of benign samples is fixed and attack samples is arranged in a series for ten iterations. The model is trained using three classifiers: KNN, FR, and LR for DDoS and Reconnaissance attacks. The threshold comparison analysis for the first part of the experiment is provided below:

5.2.1: Performance Comparison of Reconnaissance and DDoS with fixed benign samples

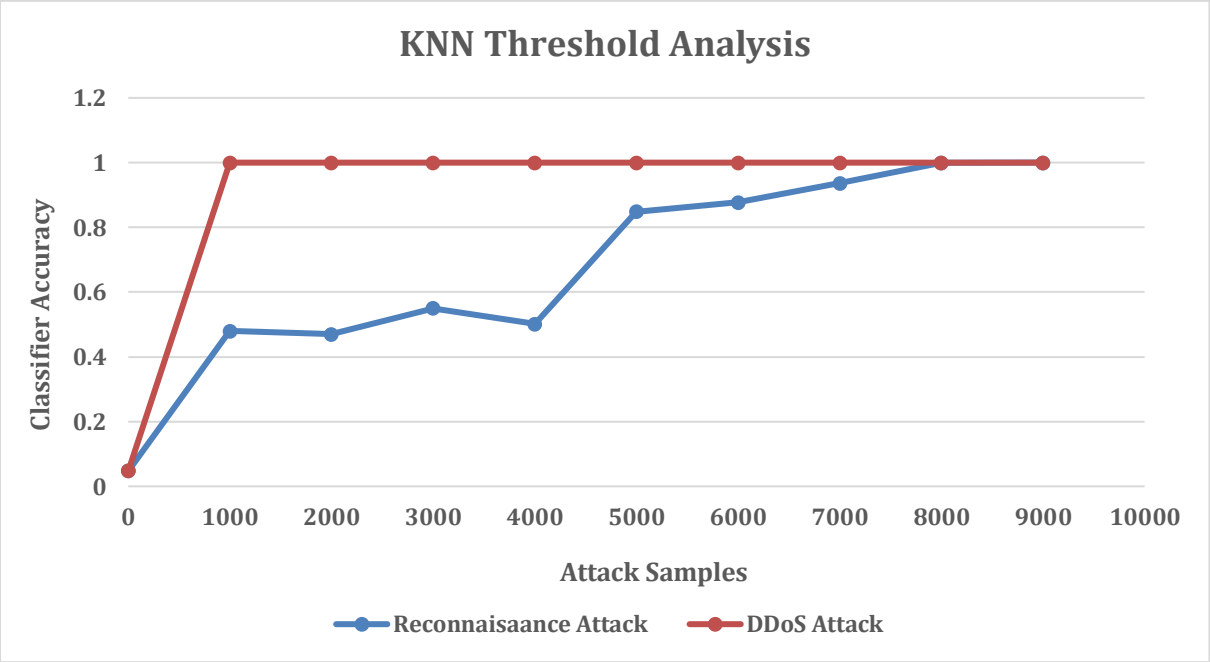

Figure 8. Threshold Analysis for KNN

The threshold analysis of KNN classifier for Reconnaissance and DDoS attacks is shown in Figure 8. The number of fixed benign samples considered for the KNN after the trial-and-error method is 500. In reconnaissance, at 7000 samples, the model attains an accuracy of 93% indicating a pre-breakthrough point for the threshold value. A threshold of 99% is achieved at 8000 attack samples for reconnaissance and 100% at 1000 samples for DDoS. The classifier accuracy in this figure shows a gradual upward trend towards the threshold value for reconnaissance whereas, in DDoS, a perfect score is achieved with 1000 attack samples. By comparing the two attacks it can be observed that DDoS requires less number of attack samples to attain the threshold value (high classifier accuracy).

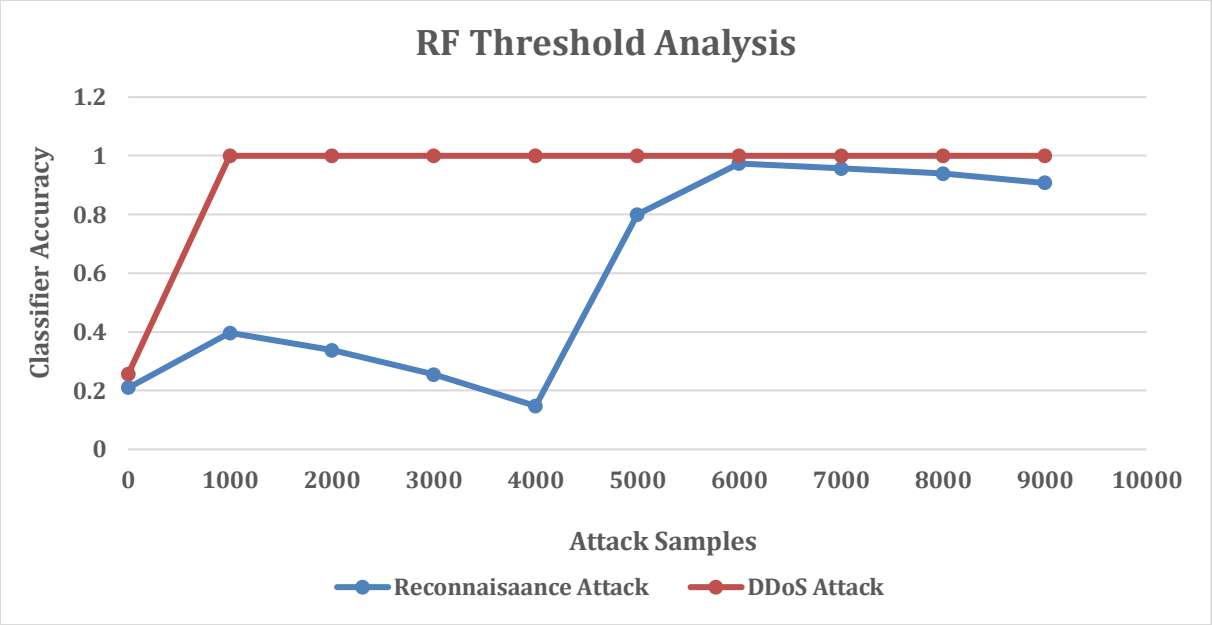

Figure 9. Threshold Analysis for RF

The threshold analysis for Random Forest classifier is depicted in Figure 9. The highest classifier score achieved with the same number of fixed benign samples (500 samples) as KNN was around 50%. This led to another round trial and error method by which the new fixed benign samples were obtained. For RF, 58 benign samples was taken as a fixed value and similar to KNN the attack samples were changed in a series for ten iterations.

The iterations executed using the new benign sample was able to achieve the threshold value of 97% using 6000 attack samples for reconnaissance. DDoS attained a threshold of 100% at 1000 samples. For reconnaissance, a very small percentage decrease is observed after the threshold iteration which was not the case for the DDoS attack. It can be noticed that similar to KNN, DDoS requires a small number of attack samples to achieve a high classifier accuracy (threshold).

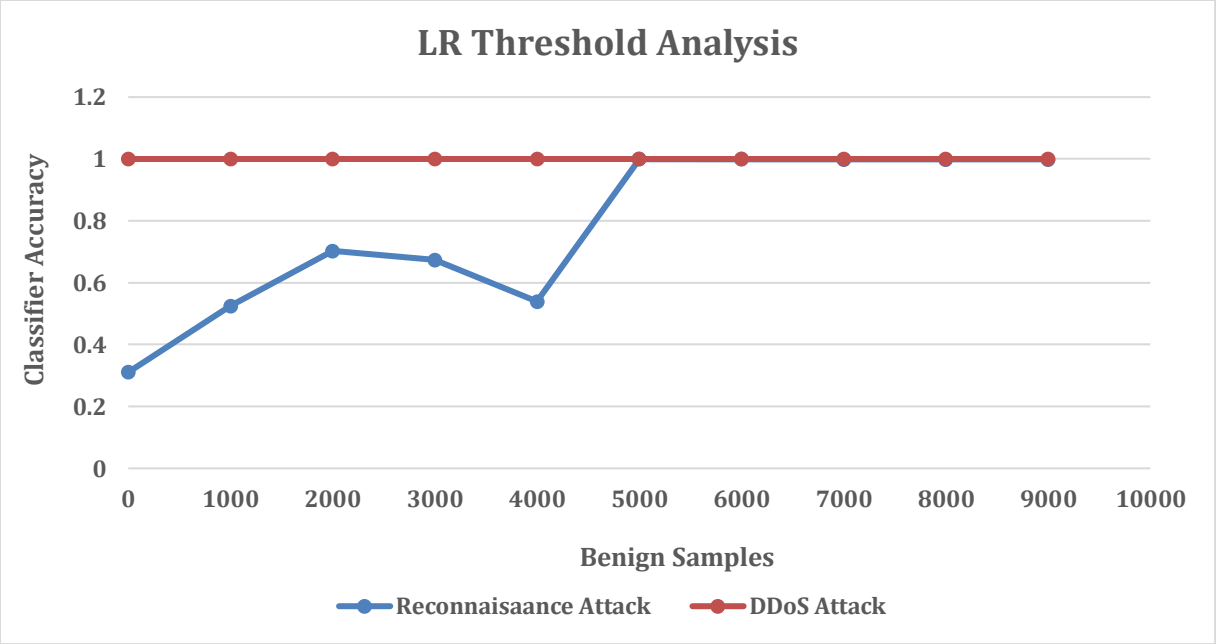

Figure 10. Threshold Analysis for LR

The threshold analysis for Logistic Regression is depicted in Figure 10 where the number of fixed benign samples taken was 500, same as KNN. The figure shows that reconnaissance achieves a threshold of 99% with 5000 attack samples and DDoS with just 1 attack sample. In Reconnaissance, a microscopic decrease in classifier score is observed after the threshold values which is not the case for DDoS attack. By comparing both types of attack in LR, DDoS requires a small number of samples to achieve the threshold.

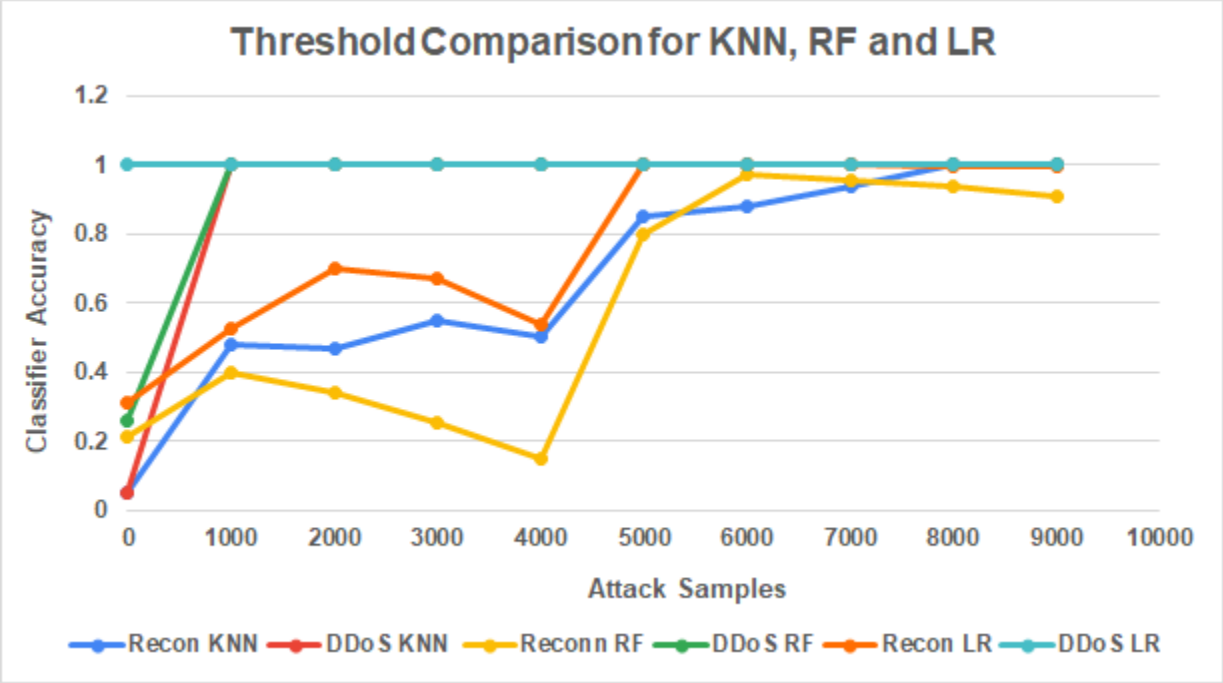

Figure 11. Comparison of Threshold Limit for KNN, RF, and LR

A combined threshold comparison for two attacks using all three classifiers is shown in Figure 11. For Reconnaissance, LR required 5000 samples to attain the threshold followed by 6000 and 8000 samples for RF and KNN respectively. In DDoS, LR was able to reach the threshold of 100% in the first iteration. KNN and RF showed similar behavior and attained the threshold at 1000 attack samples. Overall, it can be concluded from this experiment that DDoS requires less number of samples than reconnaissance to achieve the threshold for KNN, RF, and LR.

5.2.2: Performance Comparison of Reconnaissance and DDoS with fixed attack samples

In the second part of this experiment, the threshold samples for each classifier from the above analysis is taken as the starting reference iteration. In this case, attack samples are fixed and the number of benign samples is reduced in a series of ten iterations and the model is evaluated

using KNN, RF, and LR classifiers. This experiment is performed to determine the effectiveness of the threshold analysis.

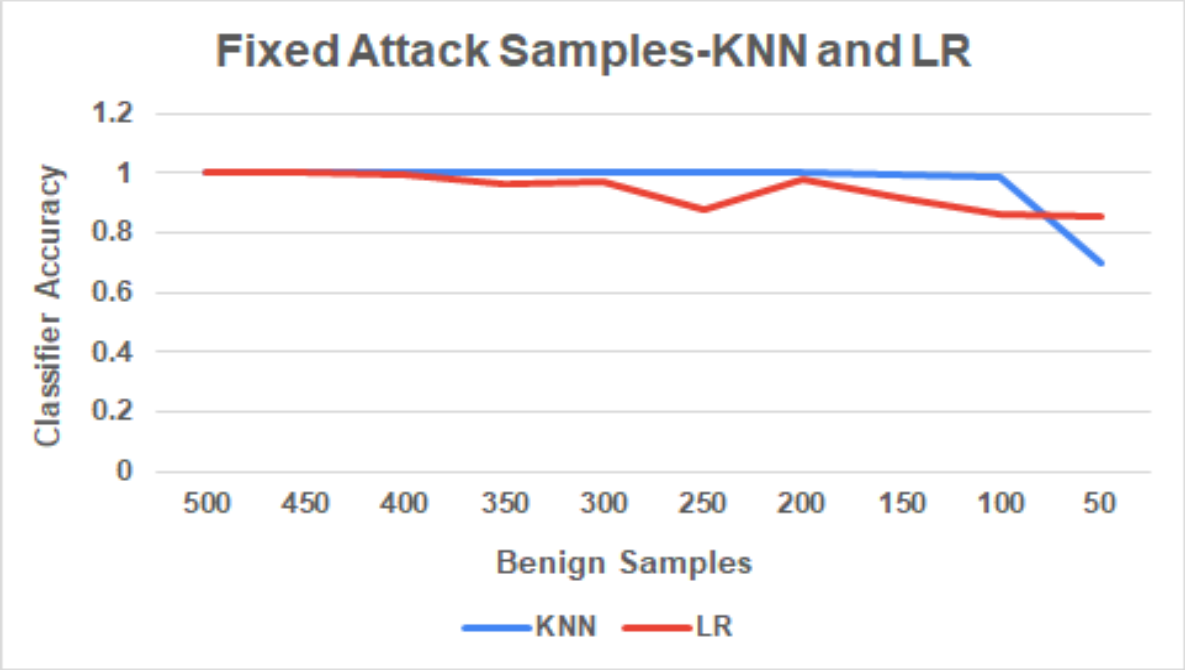

Figure 12. Part 2 analysis for KNN and LR with fixed attack samples

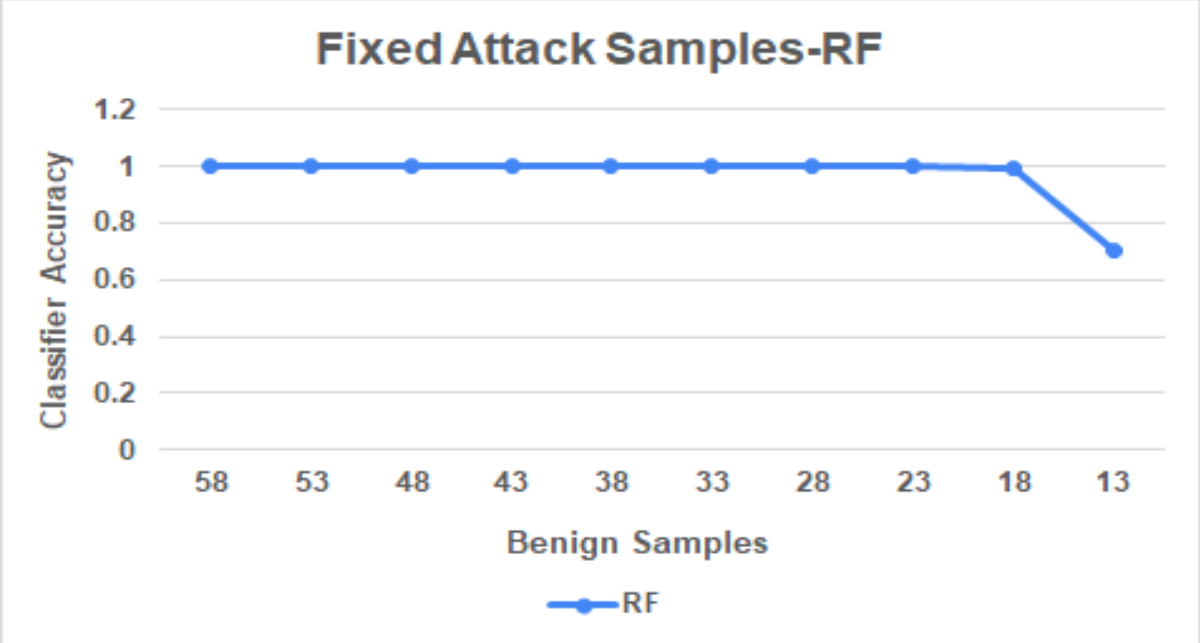

Figure 13. Part 2 analysis for RF with fixed attack samples

Figure 12 shows the KNN and LR classifier accuracy trend for reconnaissance where the number of attack samples are fixed: 8000 for KNN and 5000 for LR. In both KNN and LR, the benign samples taken for each iteration is the same i.e., benign samples are reduced with a difference of 50 from the iteration value.

In KNN, a gradual decrease is observed till 100 benign samples (9th Iteration) after which there is a 28% drop in accuracy for 50 benign samples (10th Iteration). LR shows a decreasing pattern till 250 samples (6th Iteration) followed by an increase for the 7th iteration after which the accuracy decreases once again till the very last iteration (50 benign samples).

The classifier score for the RF classifier is presented in Figure 13. In RF, the number of benign iteration samples taken is different from that of KNN and LR due to the starting threshold reference. It can be seen from the figure that there is a gradual decrease in the classifier accuracy with each iteration.

It can be concluded from the above observations that the classifier accuracy for all three classifiers decreases with the decrease in the number of samples. This study helps justify the samples taken for iterations cycles in the threshold experiment.

5.2.3: Performance Metric Analysis for Threshold Iteration Cycles

In the above analysis, the threshold limit was determined by using the classifier accuracy. This study analyzes the performance metric values of the part 1 iterations (fixed benign samples) to cross-check with previously attained threshold samples. The model was evaluated based on four performance metrics namely: Accuracy, Precision, Recall, and F1-score. The metric values for both Reconnaissance and DDoS using KNN, RF, and LR is provided below:

Table 6. Performance Metric Values for Reconnaissance attack

Iterations for Reconnaissance Attack	KNN				RF				LR			
	Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score
Iteration 1	0.05	0.025	0.5	0	0.21	0.55	0.57	0.24	0.31	0.53	0.64	0.43
Iteration 2	0.48	0.545	0.725	0.62	0.4	0.57	0.665	0.5	0.52	0.55	0.75	0.66
Iteration 3	0.47	0.55	0.72	0.61	0.34	0.57	0.63	0.41	0.7	0.58	0.84	0.81
Iteration 4	0.55	0.565	0.76	0.68	0.25	0.57	0.575	0.27	0.67	0.585	0.825	0.79
Iteration 5	0.5	0.565	0.73	0.63	0.15	0.57	0.505	0.03	0.54	0.57	0.75	0.67
Iteration 6	0.85	0.685	0.915	0.91	0.8	0.715	0.865	0.87	1	0.995	1	1
Iteration 7	0.88	0.74	0.93	0.93	0.97	0.98	0.935	0.98	1	0.995	1	1
Iteration 8	0.94	0.845	0.965	0.96	0.95	0.975	0.91	0.97	1	0.995	1	1
Iteration 9	1	1	1	1	0.94	0.96	0.905	0.96	1	0.995	1	1
Iteration 10	1	1	1	1	0.91	0.925	0.905	0.92	1	1	1	1

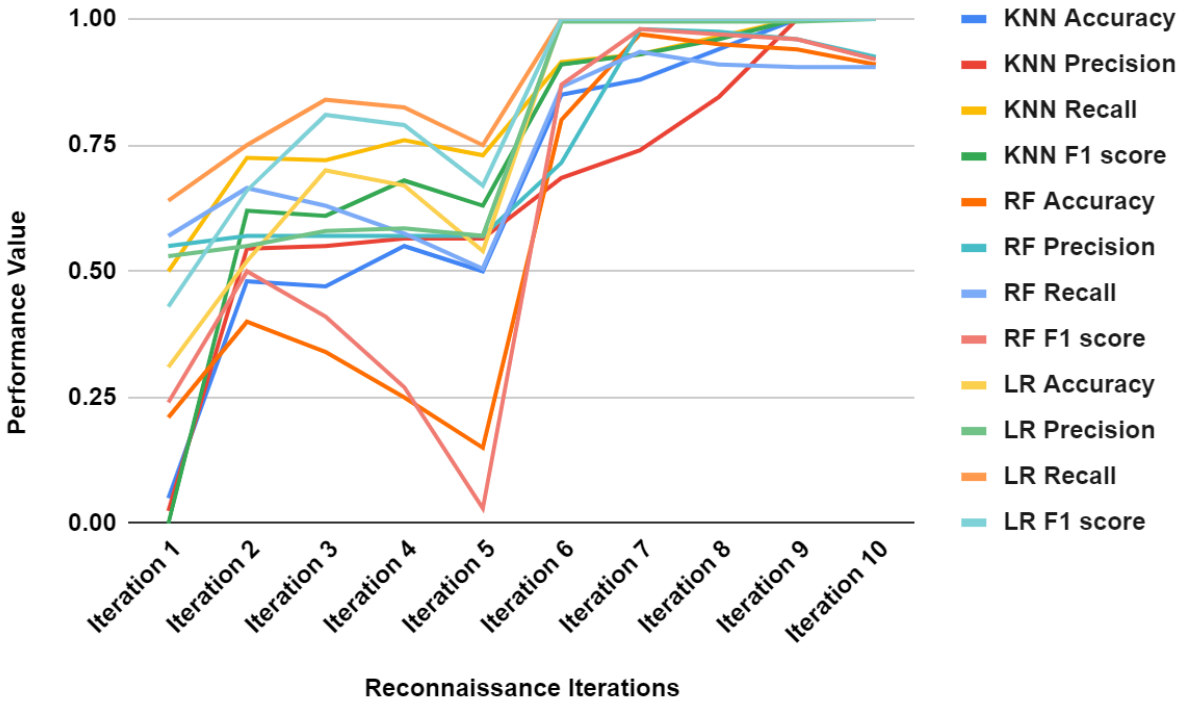

Figure 14. Comparison of performance metrics for KNN, RF, and LR-Reconnaissance

Table 6 shows the performance values of accuracy, precision, recall, and F1-score values for the Reconnaissance attack using KNN, RF, and LR classifiers. Figure 14 provides a visual representation of the same where it can be observed that the performance of all three classifiers starts to increase after Iteration 5. KNN and LR achieve 100% for all metrics in Iteration 9 and Iteration 10 respectively whereas RF attains the highest 92% for precision and F1 score, 91%

accuracy, and 90% for recall at Iteration 10. These results prove that performance metric values obtained match with that of the threshold value determined using the classifier score.

Table 7. Performance Metric Values for DDoS attack

Iterations for DDoS Attack	KNN				RF				LR			
	Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score
Iteration 1	0.05	0.025	0.5	0.045	0.26	0.55	0.595	0.255	1	1	1	1
Iteration 2	1	1	1	1	1	1	1	1	1	1	1	1
Iteration 3	1	1	1	1	1	1	1	1	1	1	1	1
Iteration 4	1	1	1	1	1	1	1	1	1	1	1	1
Iteration 5	1	1	1	1	1	1	1	1	1	1	1	1
Iteration 6	1	1	1	1	1	1	1	1	1	1	1	1
Iteration 7	1	1	1	1	1	1	1	1	1	1	1	1
Iteration 8	1	1	1	1	1	1	1	1	1	1	1	1
Iteration 9	1	1	1	1	1	1	1	1	1	1	1	1
Iteration 10	1	1	1	1	1	1	1	1	1	1	1	1

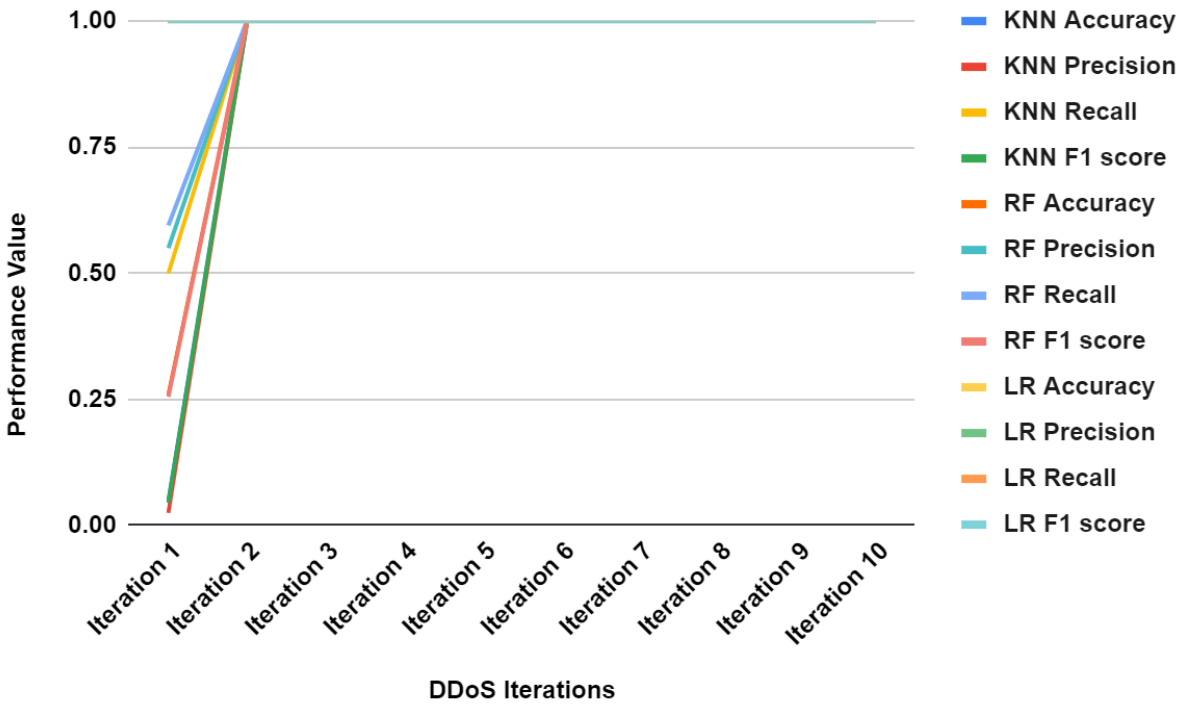

Figure 15. Comparison of performance metrics for KNN, RF, and LR-DDoS

Table 7 and Figure 15 show the values and trend of the performance metric values for the DDoS attack. It can be observed that KNN and RF achieve 100% for all metrics after Iteration 2 and LR is able to achieve perfect a value in the very first iteration.

By comparing the performance values for both Reconnaissance and DDoS, it can be concluded that DDoS can accomplish 100% performance across all four metrics in a smaller number of iterations.

5.3 Experiment 3: Feature Drop Analysis

This experiment is implemented to determine how each of the 24 preprocessed features of the BoT-IoT dataset impacts the performance of the ML model. This is implemented in 2 ways: Independent and Group. Both techniques of feature drop selection is executed on a model by using the threshold classifier accuracy as a reference (Experiment 2).

5.3.1 Independent Feature Drop

The first method of feature drop is called independent feature drop. Figure 16 shows the process where each feature is dropped individually in order, independent of other features. After each feature is dropped, the classifier accuracy value is noted and compared with the threshold value of each classifier. This comparison analysis will help determine which feature for which classifier has the biggest impact on the performance of the model.

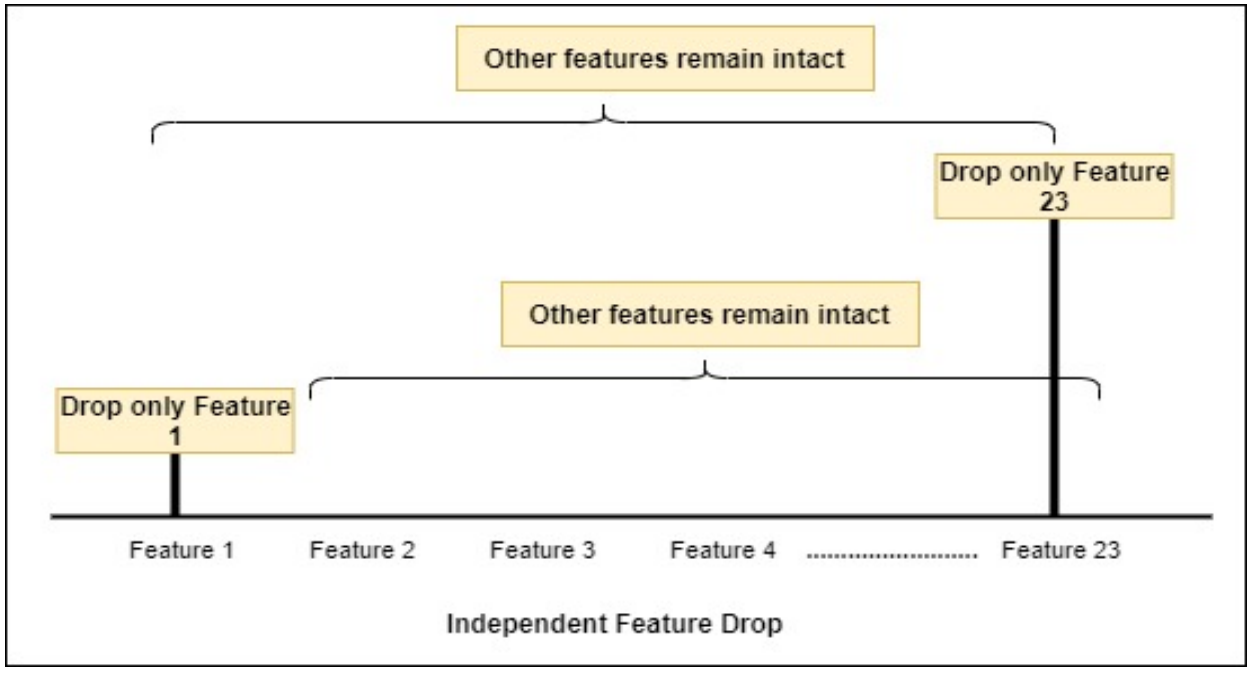

Figure 16. Independent Feature Drop

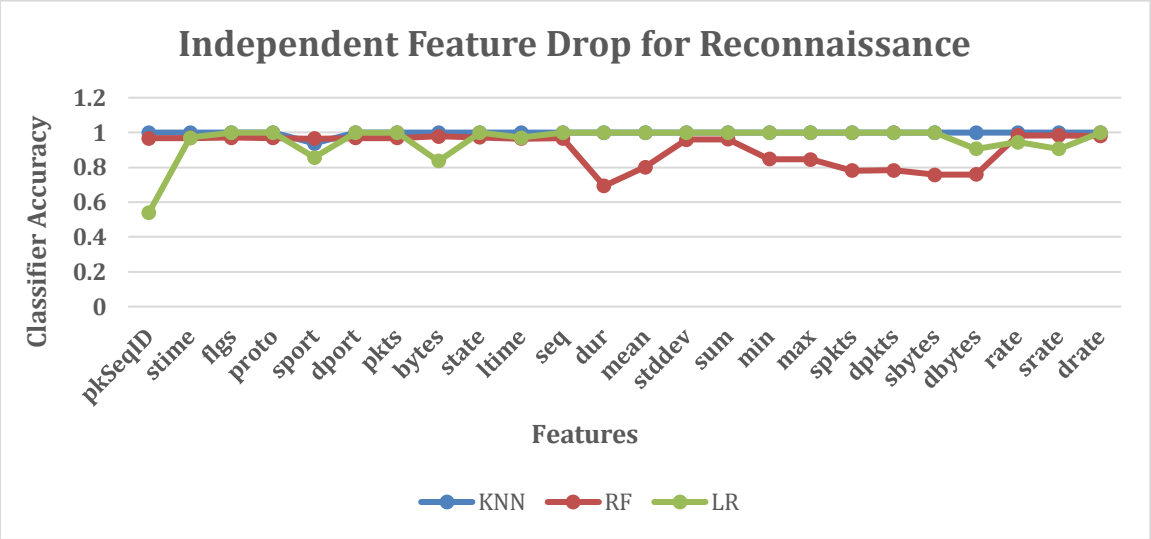

Figure 17.Independent Feature Drop for Reconnaissance attack

The reference threshold value used for the Reconnaissance attack is retrieved from Experiment 2. The independent feature drop performance of KNN, RF, and LR classifier is shown in Figure 17. In KNN, **'sport'** is the only feature that has a drop of 6% from the threshold. Other features show no change from the threshold value.

Unlike KNN, in RF, multiple features show a deviation from the threshold value. The biggest drop is observed for the **'dur'** feature (28% drop) followed by **'spkts'**, **'dpkts'**, **'sbytes' 'dybtes'** (19-20% drop), and **'mean'** (17% drop). Three features **'rate'**, **'srate',** and **'drate'** show a 0.1% increase than the threshold value. The rest of the features show an extremely small to no percentage drop in the threshold.

In LR, three features show a drop from the threshold value. **'pkSeqID'** feature showed a 45% drop followed by **'bytes'** and **'sport'** with 16% and14% drop respectively from the threshold value. Similar to RF, a small to no change in accuracy was observed for the rest of the features**.**

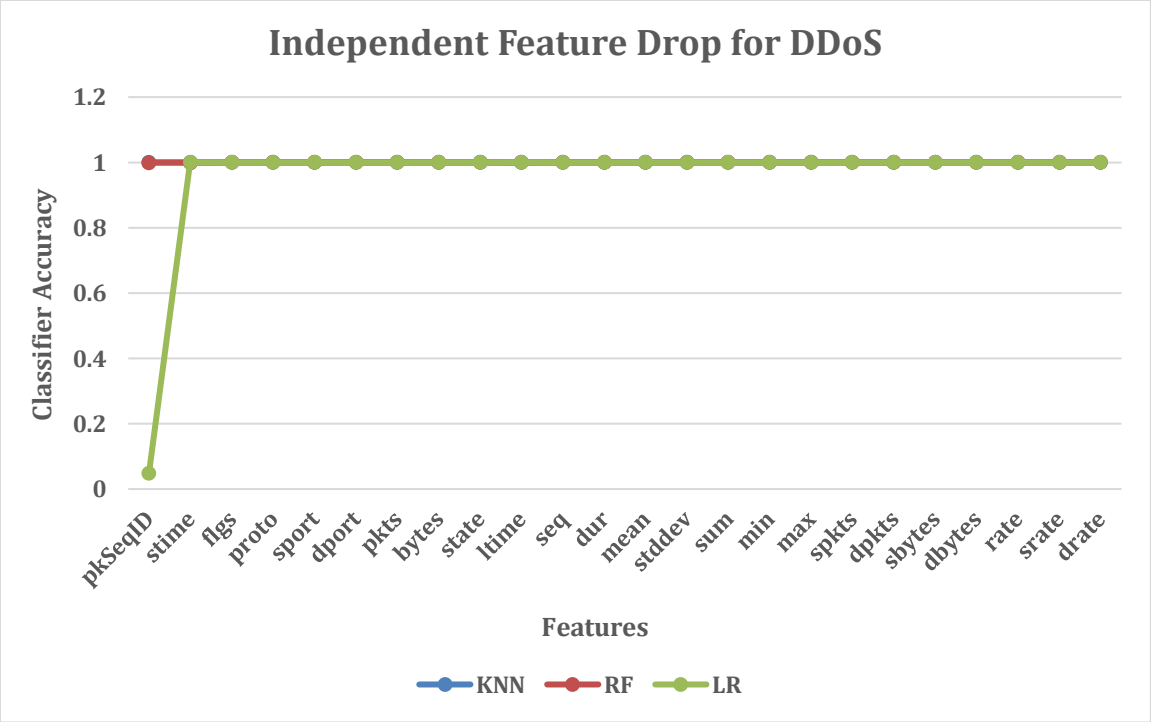

Figure 18.Independent Feature Drop for DDoS attack

Figure 18 shows the independent feature drop for DDoS attack. In KNN, no drop was detected for all the features. The features **'pkSeqID'** and **'stime'** had a 1% drop from the threshold values for RF. In LR, only **'pkSeqID'** showed a major drop, and no change in percentage was observed for other features.

5.3.2 Group Feature Drop

In the group feature drop experiment as shown in Figure 19, features are dropped in an order one after the other as a group. Similar to independent drop, the classifier score is compared to the threshold value to determine the percentage of the impact that the feature has on the performance of the model.

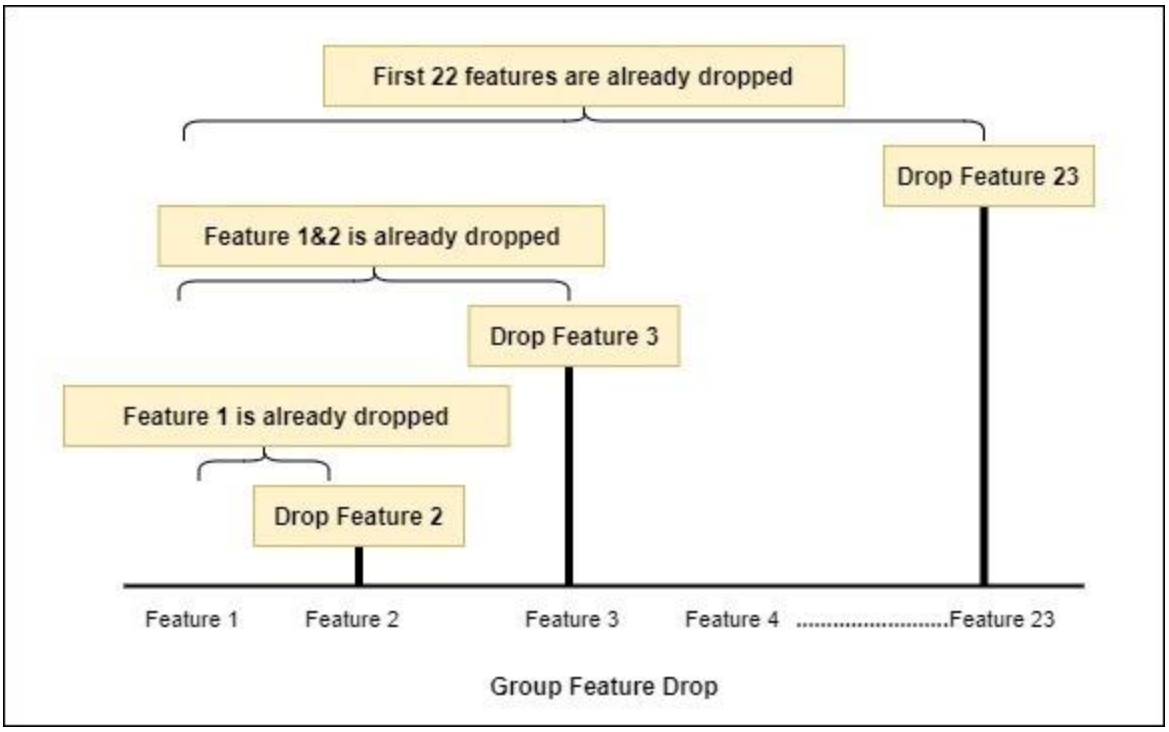

Figure 19. Group Feature Drop

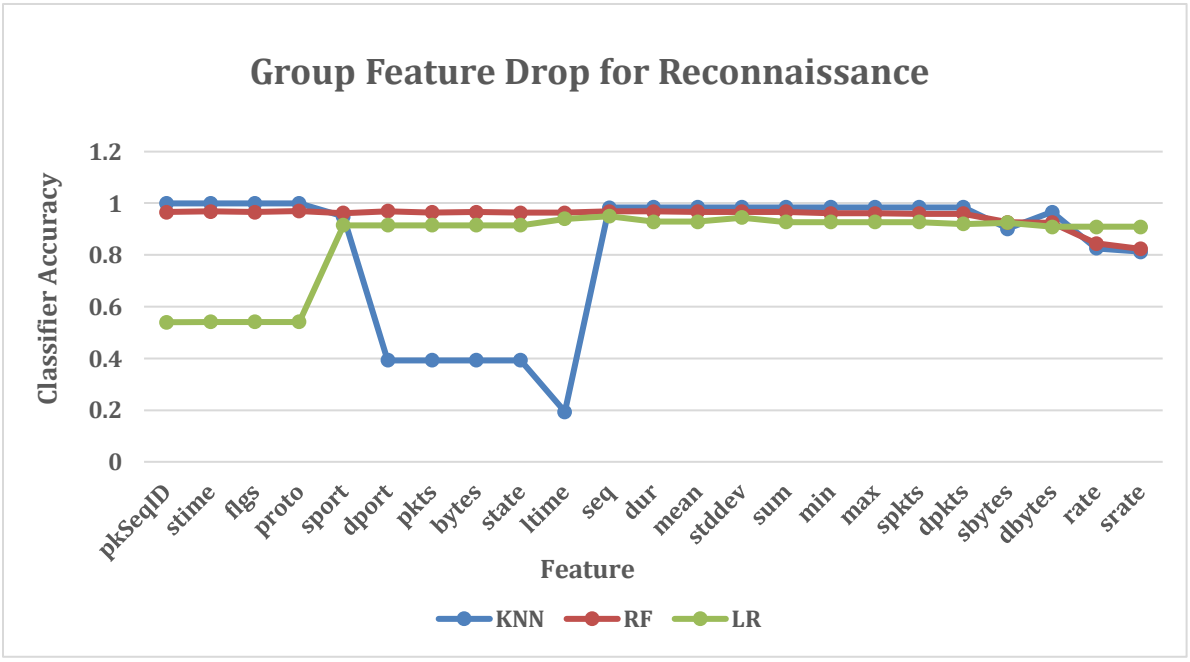

Figure 20. Group Feature Drop for Reconnaissance attack

The group feature drop values for the Reconnaissance attack is shown in Figure 20. Features were dropped one after the other as a group in order and classifier accuracy was obtained using KNN, RF, and LR classifiers. By comparing the feature drop score to the threshold value, it

can be observed that in KNN, feature **'dport'**, **'pkts'**, **'bytes'** and **'state'** showed a 60% drop (first 5 features are already dropped). **'ltime'** feature revealed an 80% drop (first 9 features already dropped).

In RF,**' rate'** and **'state'** (last two features) showed a 13-15% drop after dropping the first 20 to 21 features in order. A significant drop of around 45 % was observed for the first four features **'pkSeqID'**, **'sime**', **'flgs'**, and **'proto'** in LR.

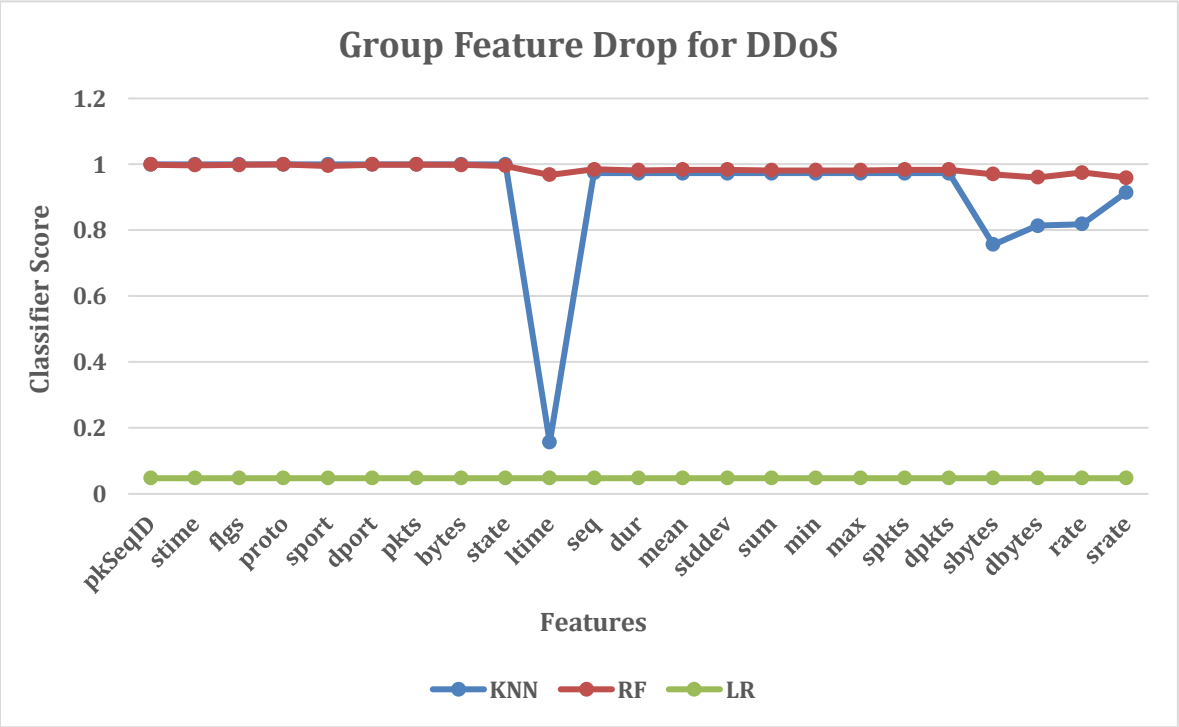

Figure 21. Group Feature Drop for DDoS attack

The group feature drop trend line for DDoS is demonstrated in Figure 21 where three features: **'ltime'**, **'sbytes**' and **'dbytes'** exhibited the drop of 85%, 25%, and 19% respectively (assuming all the features above it were already dropped). The biggest drop in RF was **'srate'** (5%) and in LR all features showed the exact score (**45% drop)** as that of LR for Reconnaissance attack.

5.3.3 Comparison between Independent and Group Feature Drop

This study provides a comparison between the two methods of feature drop techniques for each classifier and attack. It is important to note that the following observations with respect to how many features can be dropped and still maintain high model performance is applicable provided it is dropped in a particular order. Changing the order of dropping features will fetch different accuracy results for both methods of feature drop.

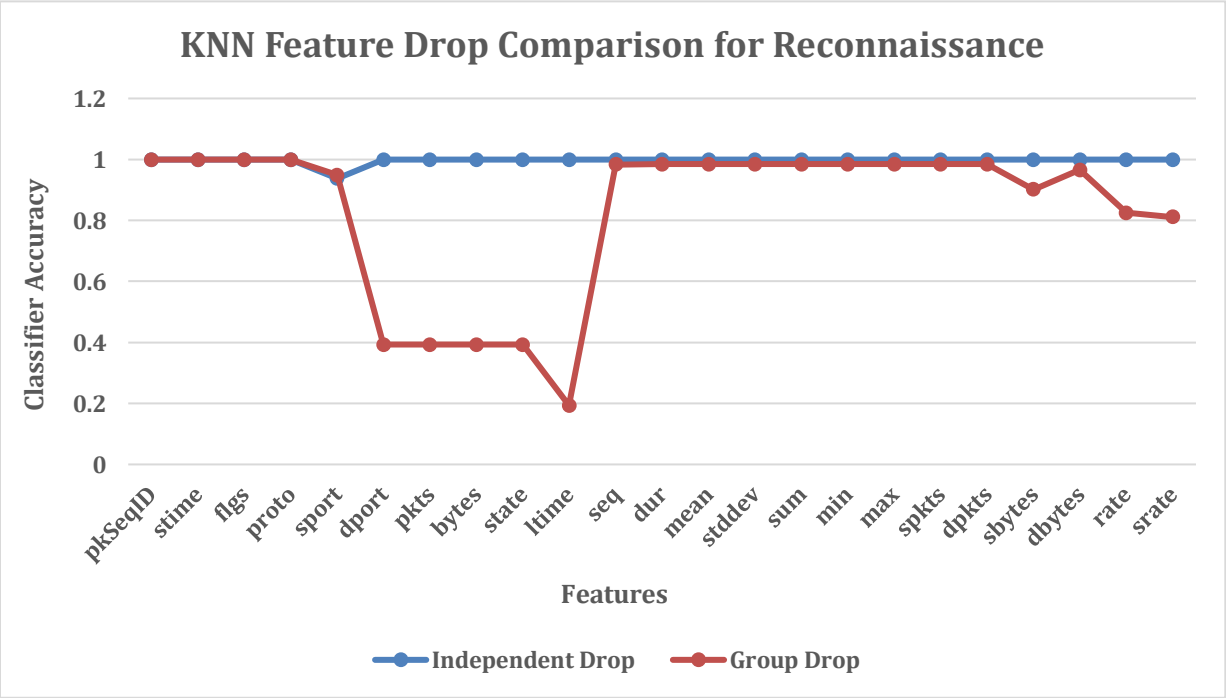

Figure 22. Feature Drop Comparison for Reconnaissance using KNN classifier

Figure 22 provides the comparison of the independent and group feature drop for reconnaissance attack using KNN classifier. It can be observed that in both techniques, **'sport'** shows a 5% drop in performance. **'dport'**, **'pkts'**, **'bytes'** **'state'** and **'ltime'** are the five features that show the biggest drop for group drop which is not the case for the independent drop.

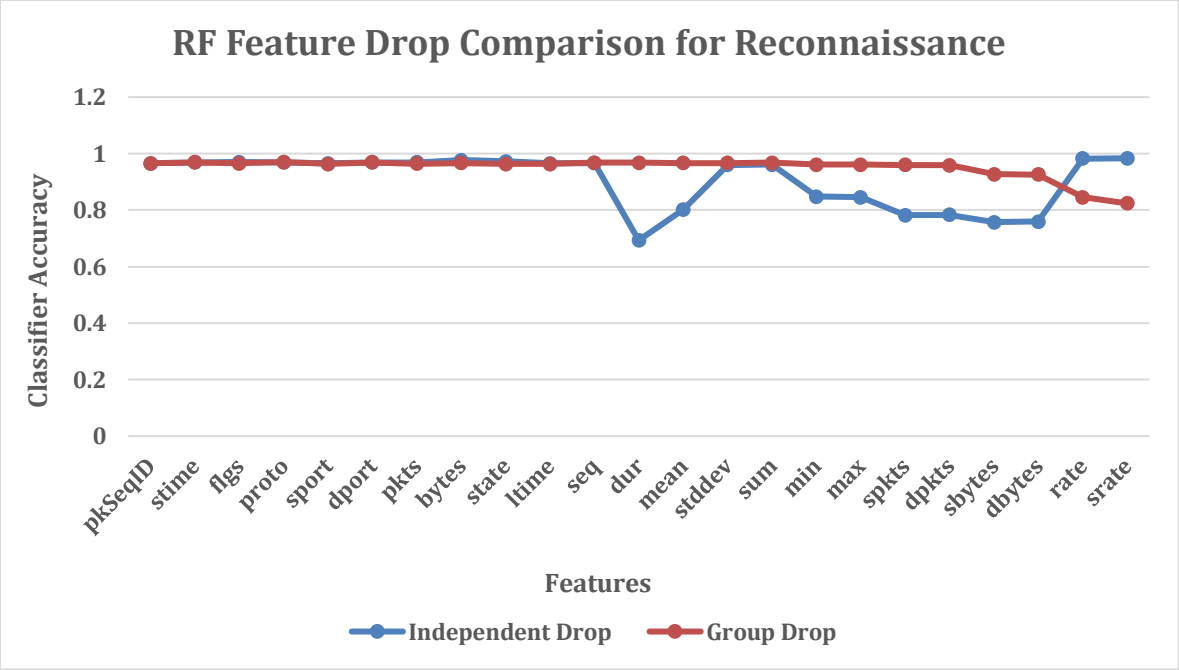

Figure 23. Feature Drop Comparison for Reconnaissance using RF classifier

The feature drop comparison trend using RF classifier for Reconnaissance attack is depicted in Figure 23, where **'seq'** is the only feature that shows a 1% drop in performance for both methods of feature drop.

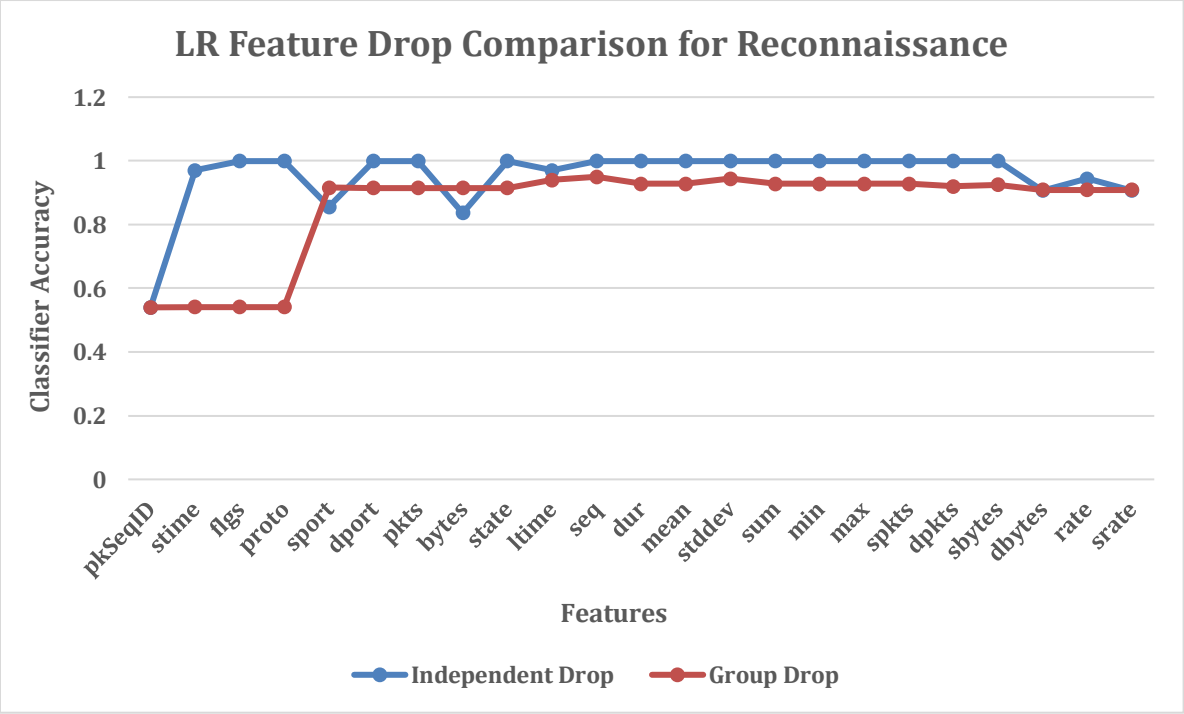

Figure 24. Feature Drop Comparison for Reconnaissance using LR classifier

Figure 24 shows the feature drop using LR classifier for reconnaissance attack where it can be noticed that classifier accuracy for the feature **'pkSeqID'** is the same for both methods, showing the biggest drop of 45%.

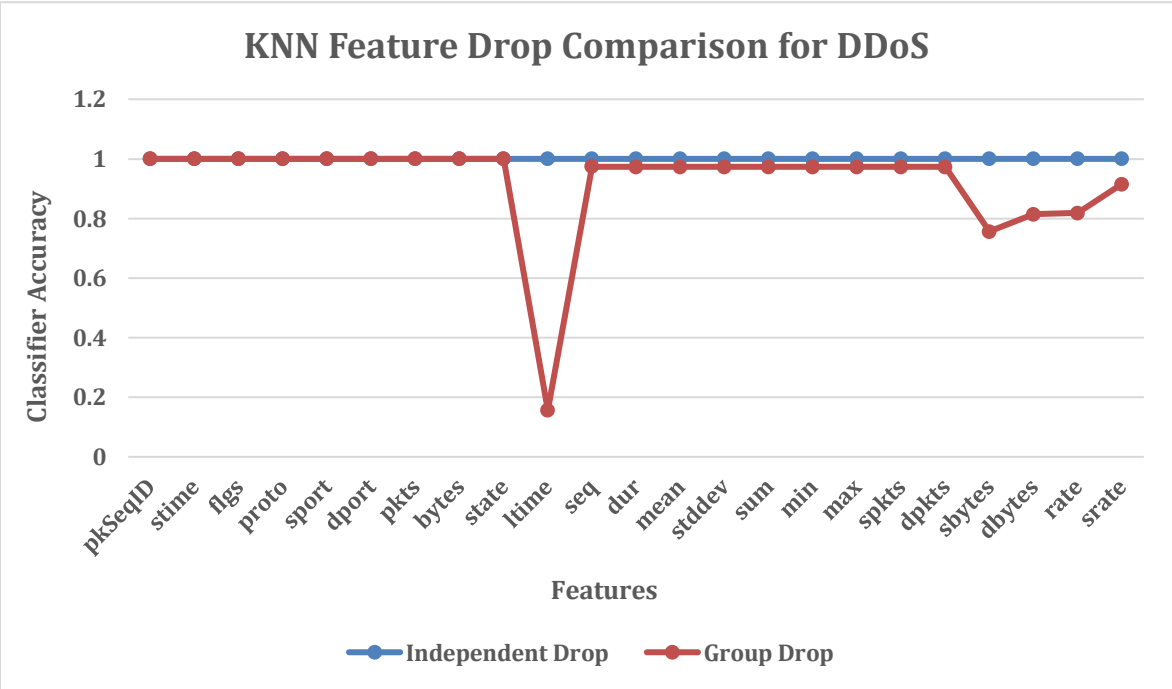

Figure 25. Feature Drop Comparison for DDoS attack using KNN classifier

The two methods of feature drop for DDoS attack using KNN classifier are shown in Figure 25. Feature **'ltime'** shows the biggest drop of 75% (first 9 features are already dropped) for the group drop method whereas there is no drop in classifier accuracy for the independent drop.

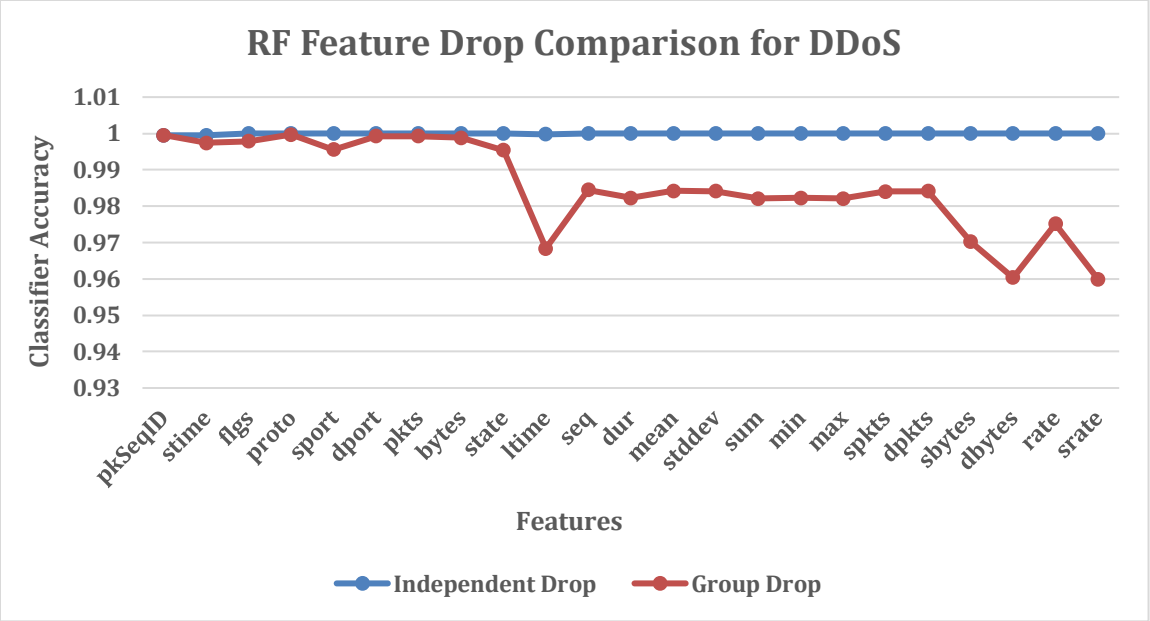

Figure 26. Feature Drop Comparison for DDoS attack using RF classifier

Figure 26 depicts the comparison for RF classifier for DDoS attack. A major drop is observed in the second half of the features for the group drop method whereas a small to no drop in performance is observed for the independent drop method.

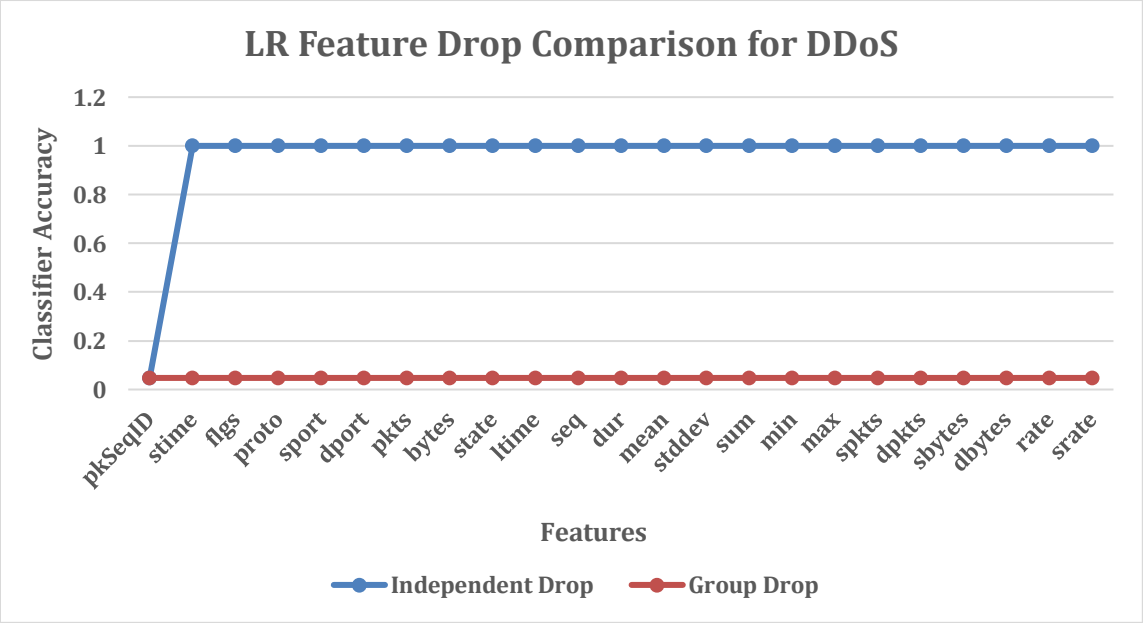

Figure 27. Feature Drop Comparison for DDoS attack using LR classifier

In LR, the same drop value is observed in both methods for the feature **‘pkSeqID’.** As shown in Figure 27. rest of the features in the group drop method have the same value as that of pkSeqID and threshold value in the case of the independent drop method.

Chapter 6

Conclusion and Future Works

In this thesis, four major findings are presented. The first finding of this work clearly indicates that there is a lack of unanimity on the techniques used in balancing datasets. This is supported by the evidence presented in this work for the formal definitions of balancing level techniques which form the basis of the taxonomy proposed here. It can be identified that these definitions are supported by the published work in two domains: related study of dataset balancing and work related to the BoT-IoT dataset.

The second finding of this study showed that the model was able to achieve a high performance despite the dataset being highly imbalanced. The ML model produced exceptional results on large number of imbalanced samples. This led to the investigation of ML model performance by using a small subset of samples. Training samples were chosen by a trial-error method for a series of ten iterations to detect the threshold sample limit at which the model achieves the highest performance. The third finding demonstrated that DDoS required lesser number of samples than Reconnaissance to achieve threshold.

The last experiment was to investigate the impact of each dataset feature on the performance of the model. This was evaluated using an independent and group feature drop method. The fourth finding of this study depicted that a certain number of features had varying impacts on the threshold value for each classifier.

For future research, there are several potential directions. One such direction that would serve as an extension of this thesis is to investigate the performance of the ML model by using different subset combinations. Another direction of potential research would be to perform other statistical techniques like Principal Component Analysis (PCA), Factor Analysis, Linear Discriminant Analysis and compare those results against the manual feature drop results of this

thesis. The last future research direction is to compare the impact of performance with and without balancing the dataset.

Bibliography

[1] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[2] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "RDTIDS: Rules and decision tree-based intrusion detection system for internet-of-things networks," *Futur. Internet*, vol. 12, no. 3, pp. 1–15, 2020, doi: 10.3390/fi12030044.

[3] M. H. Ali, M. Fadlizolkipi, A. Firdaus, and N. Z. Khidzir, "A hybrid Particle swarm optimization -Extreme Learning Machine approach for Intrusion Detection System," *2018 IEEE 16th Student Conf. Res. Dev. SCOReD 2018*, pp. 2018–2021, 2018, doi: 10.1109/SCORED.2018.8711287.

[4] L. Haripriya and M. A. Jabbar, "Role of Machine Learning in Intrusion Detection System: Review," *Proc. 2nd Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2018*, no. Iceca, pp. 925–929, 2018, doi: 10.1109/ICECA.2018.8474576.

[5] A. Haider, M. A. Khan, A. Rehman, M. Ur Rahman, and H. S. Kim, "A real-time sequential deep extreme learning machine cybersecurity intrusion detection system," *Comput. Mater. Contin.*, vol. 66, no. 2, pp. 1785–1798, 2020, doi: 10.32604/cmc.2020.013910.

[6] Kunal and M. Dua, "Machine Learning Approach to IDS: A Comprehensive Review," *Proc. 3rd Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2019*, pp. 117–121, 2019, doi: 10.1109/ICECA.2019.8822120.

[7] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," *Proc. ACM Conf. Comput. Commun. Secur.*, pp. 587–601, 2017, doi: 10.1145/3133956.3134077.

[8] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *arXiv*, 2019.

[9] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming, "AD-IoT: Anomaly detection of IoT cyberattacks in smart city using machine learning," *2019 IEEE 9th Annu. Comput. Commun. Work. Conf. CCWC 2019*, pp. 305–310, 2019, doi:

10.1109/CCWC.2019.8666450.

[10] N. Koroniotis, N. Moustafa, and E. Sitnikova, “A new network forensic framework based on deep learning for Internet of Things networks: A particle deep framework,” *Futur. Gener. Comput. Syst.*, vol. 110, pp. 91–106, 2020, doi: 10.1016/j.future.2020.03.042.

[11] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, “Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset,” *Futur. Gener. Comput. Syst.*, vol. 100, pp. 779–796, 2019, doi: 10.1016/j.future.2019.05.041.

[12] “The BoT-IoT Dataset.” [Online]. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php. [Accessed: 28-Dec-2020].

[13] “What is a Botnet Attack – Definition | Akamai.” [Online]. Available: https://www.akamai.com/uk/en/resources/what-is-a-botnet.jsp. [Accessed: 25-Dec-2020].

[14] “What is a denial of service attack (DoS) ? - Palo Alto Networks.” [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-a-denial-of-service-attack-dos. [Accessed: 25-Dec-2020].

[15] “What is an intrusion detection system? How an IDS spots threats | CSO Online.” [Online]. Available: https://www.csoonline.com/article/3255632/what-is-an-intrusion-detection-system-how-an-ids-spots-threats.html. [Accessed: 25-Dec-2020].

[16] “What is an Intrusion Detection System (IDS)? | Check Point Software.” [Online]. Available: https://www.checkpoint.com/cyber-hub/network-security/what-is-an-intrusion-detection-system-ids/. [Accessed: 25-Dec-2020].

[17] “What Is Machine Learning: Definition, Types, Applications and Examples | Potentia Analytics Inc.” [Online]. Available: https://www.potentiaco.com/what-is-machine-learning-definition-types-applications-and-examples/. [Accessed: 25-Dec-2020].

[18] “7 Steps to Machine Learning: How to Prepare for an Automated Future | by Dr Mark van Rijmenam | DataSeries | Medium.” [Online]. Available: https://medium.com/dataseries/7-steps-to-machine-learning-how-to-prepare-for-an-automated-future-78c7918cb35d. [Accessed: 25-Dec-2020].

[19] “What is machine learning? Intelligence derived from data | InfoWorld.” [Online]. Available: https://www.infoworld.com/article/3214424/what-is-machine-learning-

intelligence-derived-from-data.html. [Accessed: 25-Dec-2020].

[20] "What is Machine Learning? A definition - Expert System | Expert.ai." [Online]. Available: https://www.expert.ai/blog/machine-learning-definition/. [Accessed: 28-Dec-2020].

[21] "6 Types of Supervised Learning You Must Know About in 2020 | upGrad blog." [Online]. Available: https://www.upgrad.com/blog/types-of-supervised-learning/. [Accessed: 25-Dec-2020].

[22] "What is Machine Learning? | Types of Machine Learning | Edureka." [Online]. Available: https://www.edureka.co/blog/what-is-machine-learning/. [Accessed: 25-Dec-2020].

[23] "7 Steps of Machine Learning." [Online]. Available: https://livecodestream.dev/post/7-steps-of-machine-learning/. [Accessed: 25-Dec-2020].

[24] "The 7 Key Steps To Build Your Machine Learning Model." [Online]. Available: https://analyticsindiamag.com/the-7-key-steps-to-build-your-machine-learning-model/. [Accessed: 28-Dec-2020].

[25] "Frameworks for Approaching the Machine Learning Process." [Online]. Available: https://www.kdnuggets.com/2018/05/general-approaches-machine-learning-process.html. [Accessed: 25-Dec-2020].

[26] "The 5 Steps of Machine Learning." [Online]. Available: https://www.linkedin.com/pulse/5-steps-machine-learning-zaynah-bhanji. [Accessed: 28-Dec-2020].

[27] "Data Preprocessing in Machine learning - Javatpoint." [Online]. Available: https://www.javatpoint.com/data-preprocessing-machine-learning. [Accessed: 25-Dec-2020].

[28] "Pre-Processing for Machine Learning Projects | Data Science and Machine Learning | Kaggle." [Online]. Available: https://www.kaggle.com/general/69712. [Accessed: 25-Dec-2020].

[29] "Data Preprocessing : Concepts. Introduction to the concepts of Data… | by Pranjal Pandey | Towards Data Science." [Online]. Available: https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825. [Accessed: 25-Dec-2020].

[30] “Data Preprocessing in Machine Learning.” [Online]. Available: https://serokell.io/blog/data-preprocessing. [Accessed: 25-Dec-2020].

[31] P. Laskov, D. Patrick, and C. Sch, “Learning Intrusion Detection : Supervised or Unsupervised ? Learning intrusion detection : supervised or unsupervised ?,” no. September 2005, pp. 50–57, 2014, doi: 10.1007/11553595.

[32] “Understanding Data Bias. Types and sources of data bias | by Prabhakar Krishnamurthy | Towards Data Science.” [Online]. Available: https://towardsdatascience.com/survey-d4f168791e57. [Accessed: 25-Dec-2020].

[33] G. Karatas, O. Demir, and O. K. Sahingoz, “Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset,” *IEEE Access*, vol. 8, pp. 32150–32162, 2020, doi: 10.1109/ACCESS.2020.2973219.

[34] B. Subba, S. Biswas, and S. Karmakar, “A Neural Network based system for Intrusion Detection and attack classification,” *2016 22nd Natl. Conf. Commun. NCC 2016*, no. May, 2016, doi: 10.1109/NCC.2016.7561088.

[35] H. Al Najada, I. Mahgoub, and I. Mohammed, “Cyber Intrusion Prediction and Taxonomy System Using Deep Learning and Distributed Big Data Processing,” *Proc. 2018 IEEE Symp. Ser. Comput. Intell. SSCI 2018*, pp. 631–638, 2019, doi: 10.1109/SSCI.2018.8628685.

[36] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, “A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks,” *Electron.*, vol. 8, no. 11, 2019, doi: 10.3390/electronics8111210.

[37] M. Asadi, M. A. Jabraeil Jamali, S. Parsa, and V. Majidnezhad, “Detecting botnet by using particle swarm optimization algorithm based on voting system,” *Futur. Gener. Comput. Syst.*, vol. 107, pp. 95–111, 2020, doi: 10.1016/j.future.2020.01.055.

[38] O. Ibitoye, O. Shafiq, and A. Matrawy, “Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks,” *arXiv*, 2019.

[39] Z. A. Baig, S. Sanguanpong, S. N. Firdous, V. N. Vo, T. G. Nguyen, and C. So-In, “Averaged dependence estimators for DoS attack detection in IoT networks,” *Futur. Gener. Comput. Syst.*, vol. 102, pp. 198–209, 2020, doi: 10.1016/j.future.2019.08.007.

[40] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study,” *J. Inf. Secur.*

Appl., vol. 50, p. 102419, 2020, doi: 10.1016/j.jisa.2019.102419.

[41] M. A. Ferrag and L. Maglaras, "DeepCoin: A Novel Deep Learning and Blockchain-Based Energy Exchange Framework for Smart Grids," *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1285–1297, 2020, doi: 10.1109/TEM.2019.2922936.

[42] J. Galeano-Brajones, J. Carmona-Murillo, J. F. Valenzuela-Valdés, and F. Luna-Valero, "Detection and mitigation of DoS and DDoS attacks in iot-based stateful SDN: An experimental approach," *Sensors (Switzerland)*, vol. 20, no. 3, pp. 1–18, 2020, doi: 10.3390/s20030816.

[43] N. Guizani and A. Ghafoor, "A network function virtualization system for detecting malware in large IoT based networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1218–1228, 2020, doi: 10.1109/JSAC.2020.2986618.

[44] M. A. Lawal, R. A. Shaikh, and S. R. Hassan, "An anomaly mitigation framework for iot using fog computing," *Electron.*, vol. 9, no. 10, pp. 1–24, 2020, doi: 10.3390/electronics9101565.

[45] M. Ge, N. F. Syed, X. Fu, Z. Baig, and A. Robles-Kelly, "Toward a Deep Learning-Driven Intrusion Detection Approach for Internet of Things," *arXiv*, 2020.

[46] S. Dwibedi, "A Comparative Study on Contemporary Intrusion Detection Datasets for Machine Learning Research," 2018.

[47] S. Liaqat, A. Akhunzada, F. S. Shaikh, A. Giannetsos, and M. A. Jan, "SDN orchestration to combat evolving cyber threats in Internet of Medical Things (IoMT)," *Comput. Commun.*, vol. 160, no. April, pp. 697–705, 2020, doi: 10.1016/j.comcom.2020.07.006.

[48] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "IoT malicious traffic identification using wrapper-based feature selection mechanisms," *Comput. Secur.*, vol. 94, 2020, doi: 10.1016/j.cose.2020.101863.

[49] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, "A survey on addressing high-class imbalance in big data," *J. Big Data*, vol. 5, no. 1, 2018, doi: 10.1186/s40537-018-0151-6.

[50] Y. Sun, A. K. C. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 23, no. 4, pp. 687–719, 2009, doi: 10.1142/S0218001409007326.

[51] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, "Classification with class imbalance

problem: A review," *Int. J. Adv. Soft Comput. its Appl.*, vol. 7, no. 3, pp. 176–204, 2015.

[52] "A Primer to Ensemble Learning – Bagging and Boosting ." [Online]. Available: https://analyticsindiamag.com/primer-ensemble-learning-bagging-boosting/. [Accessed: 31-Jul-2020].

[53] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, "Deep learning-based intrusion detection for IoT networks," *Proc. IEEE Pacific Rim Int. Symp. Dependable Comput. PRDC*, vol. 2019-Decem, pp. 256–265, 2019, doi: 10.1109/PRDC47002.2019.00056.

[54] "Battling Data Imbalance in IBM HR Attrition Challenge | by Александра Пухова | Medium." [Online]. Available: https://medium.com/@alepukhova526/battling-data-imbalance-in-ibm-hr-attrition-challenge-3a26337a4943. [Accessed: 31-Jul-2020].

[55] I. A. Jimoh, I. Ismaila, and M. Olalere, "Enhanced Decision Tree-J48 with SMOTE Machine Learning Algorithm for Effective Botnet Detection in Imbalance Dataset," *2019 15th Int. Conf. Electron. Comput. Comput. ICECCO 2019*, no. Icecco, 2019, doi: 10.1109/ICECCO48375.2019.9043233.

[56] L. McNabb and R. S. Laramee, "How to Write a Visualization Survey Paper : A Starting Point," 2019, doi: 10.2312/eged.20191026.

[57] J. Beel, B. Gipp, and E. Wilde, "Academic search engine optimization(ASEO): Optimizing Scholarly Literature for Google Scholar & Co," *J. Sch. Publ.*, vol. 41, no. 2, pp. 176–190, 2010, doi: 10.3138/jsp.41.2.176.

[58] "K-Nearest Neighbors Algorithm in Python and Scikit-Learn." [Online]. Available: https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/. [Accessed: 28-Dec-2020].

[59] "Machine Learning Basics with the K-Nearest Neighbors Algorithm | by Onel Harrison | Towards Data Science." [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. [Accessed: 28-Dec-2020].

[60] "Machine Learning Random Forest Algorithm - Javatpoint." [Online]. Available: https://www.javatpoint.com/machine-learning-random-forest-algorithm. [Accessed: 28-Dec-2020].

[61] "Logistic Regression in Machine Learning - Javatpoint." [Online]. Available:

https://www.javatpoint.com/logistic-regression-in-machine-learning. [Accessed: 28-Dec-2020].

[62] "Evaluation Metrics for Machine Learning Models | by Bhajandeep Singh | Heartbeat." [Online]. Available: https://heartbeat.fritz.ai/evaluation-metrics-for-machine-learning-models-d42138496366. [Accessed: 28-Dec-2020].

[63] "Confusion Matrix in Machine Learning - Javatpoint." [Online]. Available: https://www.javatpoint.com/confusion-matrix-in-machine-learning. [Accessed: 28-Dec-2020].

[64] M. H. Abdulraheem and N. B. Ibraheem, "A detailed analysis of new intrusion detection dataset," *J. Theor. Appl. Inf. Technol.*, vol. 97, no. 17, pp. 4519–4537, 2019.